

APPENDIX A

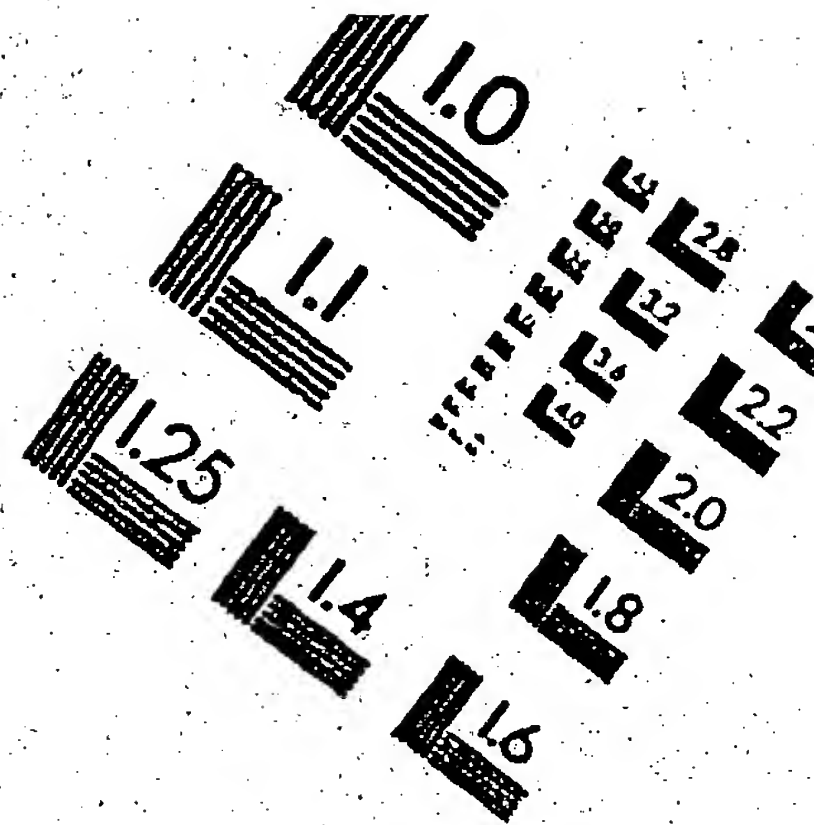
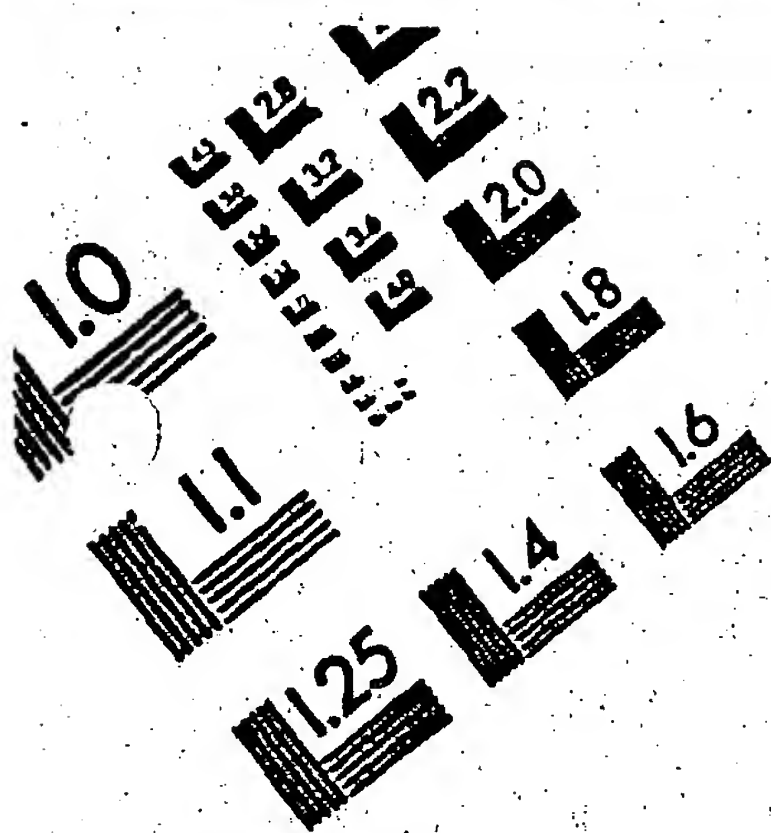
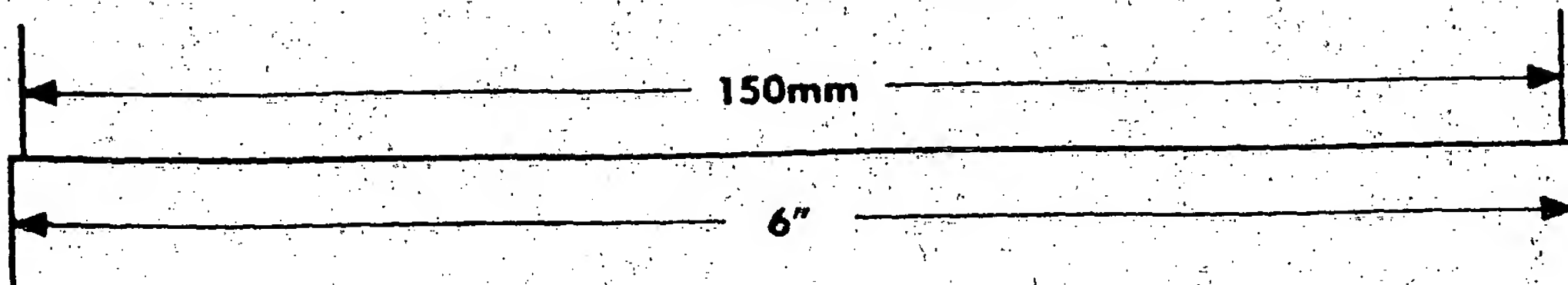
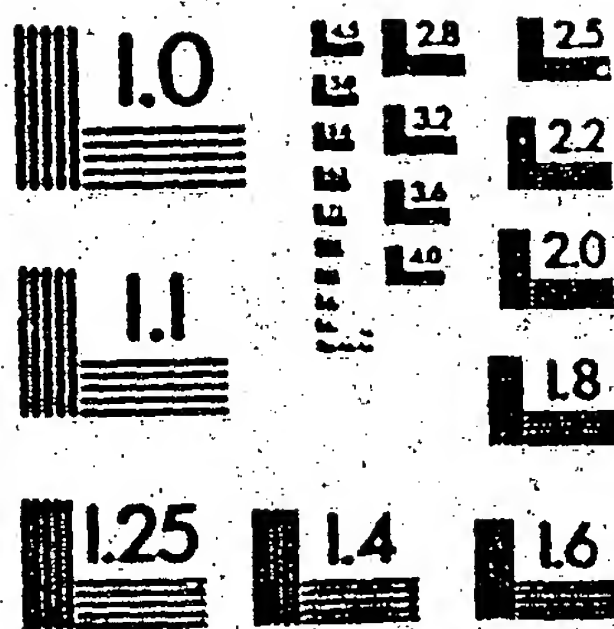
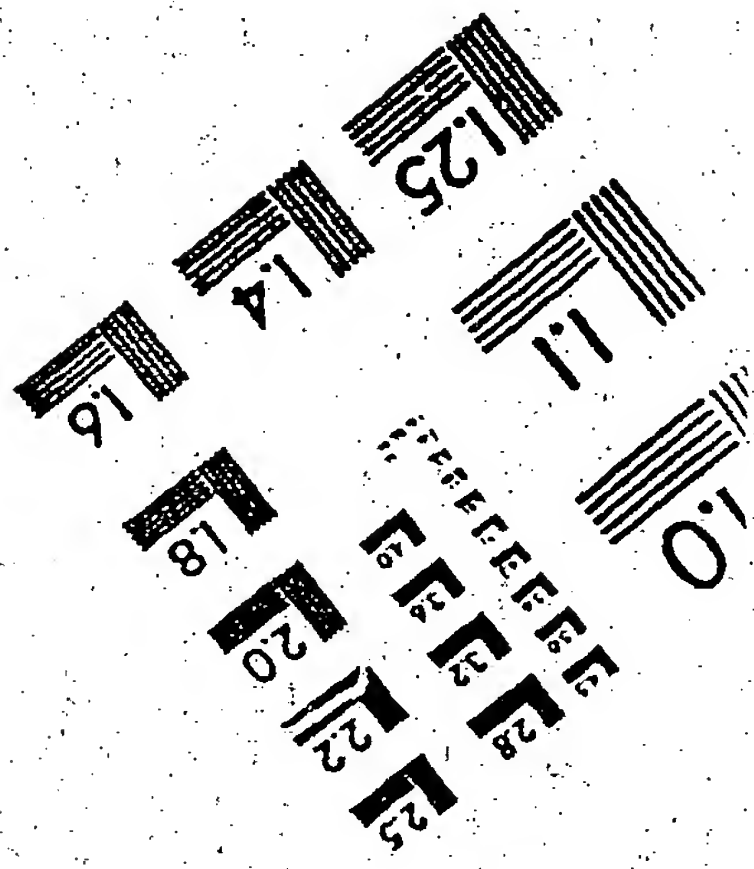
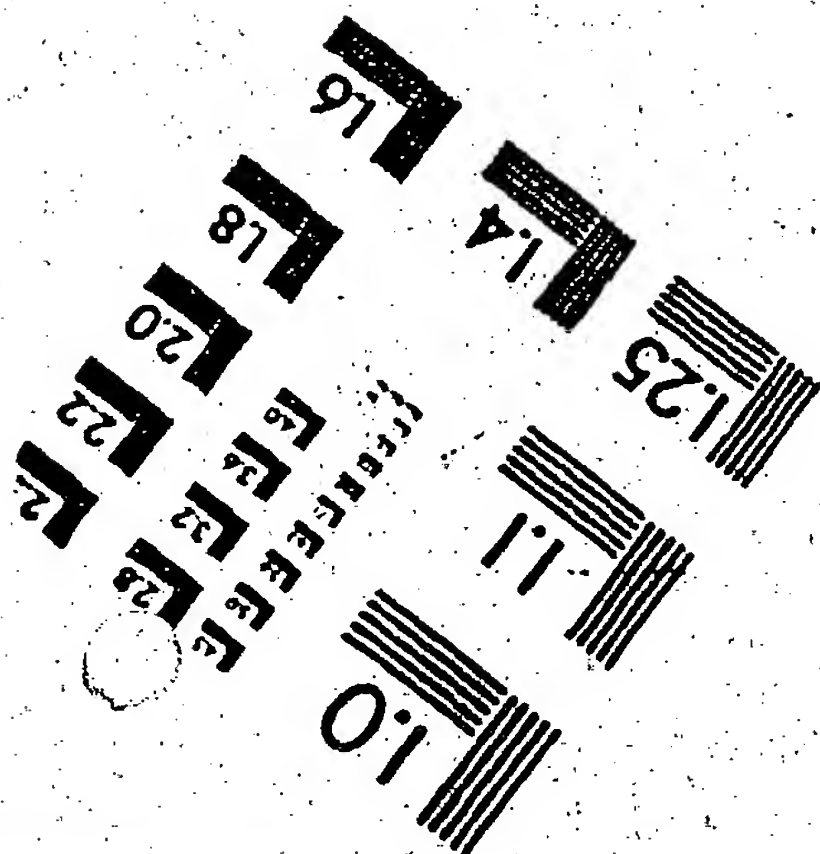


IMAGE EVALUATION TEST TARGET QA-3



APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989



Case No. 9262/3

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
APPLICATION FOR UNITED STATES LETTERS PATENT

INVENTOR:

JAN WASOWICZ

TITLE:

ADAPTIVE AUDITORY AND
PHONOLOGICAL APPARATUS AND
METHOD

MICROFICHE APPENDIX A

APPENDIX A-1

Movie Script4

```
on startMovie
  global gdataFieldSprites, gthename, grecordkeeper, Pro, gnewbie, gwhatversion, enterpasslist
  global gObPrefsMan
  put getProStatus(grecordkeeper) into Pro
  -- set pro = 1 --TTEST
  case (pro) of
    1:
      repeat with x= 44 to 48
        -- puppetsprite x, true
        set the visible of sprite x to false
      end repeat
      repeat with x = 33 to 36
        -- puppetsprite x, true
        set the visible of sprite x to true
      end repeat
    2:
      repeat with x= 44 to 48
        -- puppetsprite x, true
        set the visible of sprite x to true
      end repeat
  end case

  set enterpasslist = list()
  put "Are you sure you want to delete player" into line 1 of member "delete text"

  put "" into member "fake pass"
  -- put "" into member "enter new pass"
  -- put empty into field "enter new pass"

  -- set gdataFieldSprites = [1,2,3]
  cursor 4
  set the font of member "UserName" = "helvetica"
  set the fontsize of member "UserName" = 14
  set the forecolor of member "UserName" = the forecolor of member "color"

  set the font of member "enter name" = "helvetica"
  set the fontsize of member "enter name" = 14
  set the forecolor of member "enter name" = the forecolor of member "color"

  set the font of member "enter new pass" = "helvetica"
  set the fontsize of member "enter new pass" = 14
  set the forecolor of member "enter new pass" = the forecolor of member "pass color"

  set the font of member "confirm new pass" = "helvetica"
  set the fontsize of member "confirm new pass" = 14
  set the forecolor of member "confirm new pass" = the forecolor of member "pass color"

  set the font of member "enter pass" = "helvetica"
  set the fontsize of member "enter pass" = 14
  set the forecolor of member "enter pass" = the forecolor of member "pass color"
```

```

-- repeat with x in gDataFieldSprites
--   put " " into member x
--   set the font of member x = "Geneva"
--   set the fontsize of member x = 29
--   set the editable of sprite x = true
--   puppetsprite x ,true
-- end repeat

put gthename into field "username"

set gwhatversion = 0

put " " into field "enter name"

if not objectP(gObPrefsMan) then
  set gObPrefsMan = 0
  set gObPrefsMan = new(script "PrefsScreenManager")
end if

end

on stopmovie
  global gVoid, gObPrefsMan, gDTPopUpMan

  if the runMode <> "author" then
    set gDTPopUpMan = gVoid
    set gObPrefsMan = gVoid
  end if

  if the runmode = "author" then
    put "x" into field "UserName"
    put "x" into field "enter name"
    put "x" into field "UserNameTitle"

    put "x" into field "enter new pass"
    put "x" into field "confirm new pass"
    put "x" into field "enter pass"
    put "x" into field "fake pass"

    set the font of member "UserName" = "helvetica"
    set the fontsize of member "UserName" = 14
    set the forecolor of member "UserName" = the forecolor of member "spot 1"

    set the font of member "enter name" = "helvetica"
    set the fontsize of member "enter name" = 14
    set the forecolor of member "enter name" = the forecolor of member "spot 1"

    repeat with x = 2 to 26
      set targetfield = "spot" && x
      put " " into field targetfield
    end repeat

  end if

```

end

```
on buttonDownhandler
  put the clickon into x
  set buttonName = word 1 of the name of member (the member of sprite x)

  set downbutton = buttonName && "down"
  set upbutton = buttonName && "up"
  set the member of sprite x to member downbutton
  updatestage
```

```
  repeat while the mousedown
    if rollover(x) then
      set the member of sprite x to member downbutton
      updatestage
    else
      set the member of sprite x to member upbutton
      updatestage
    end if
  end repeat
```

end

```
on arrowuphandler
  put the clickon into x

  put word 1 of the name of member the mousecast into buttonName
  set downbutton = buttonName && "down"
  set upbutton = buttonName && "up"
  if the name of member the mousecast = downbutton then
    set the member of sprite x to member upbutton
    updatestage
    case (buttonname) of
      "leftarrow" : cursor 4
      go to marker (-1)
      "rightarrow" : cursor 4
      go to marker (+1)
    end case
  end if
```

end

```
on buttonUPhandler
  global gtheName,theGame
  put the clickon into x
```

```
  put word 1 of the name of member the mousecast into buttonName
  set downbutton = buttonName && "down"
  set upbutton = buttonName && "up"
  if the name of member the mousecast = downbutton then
```

```
set the member of sprite x to member upbutton
updatestage
```

```
-- sound stop 1
-- puppetsound 0
```

```
cursor 4
```

```
if field "UserName" <> "" then
    put field "UserName" into theName
```

```
set gtheName = theName
```

```
case (buttonname) of
```

```
    "clown" : put "5" into theGame
    repeat with x = 3 to 18
        puppetsprite x, false
    end repeat
```

```
    go to frame "mac"
    go to movie "Karlooon8"
```

```
    "coal" : put "6" into theGame
    repeat with x = 3 to 18
        puppetsprite x, false
    end repeat
```

```
    go to frame "black"
    go to movie "coal8"
```

```
    "rapper" : put "4" into theGame
    repeat with x = 3 to 18
        puppetsprite x, false
    end repeat
```

```
    go to frame "black"
    go to movie "rappers8"
```

```
    "katy" : put "1" into theGame
    repeat with x = 3 to 18
        puppetsprite x, false
    end repeat
```

```
    go to frame "black"
    go to movie "Katy8"
```

```
    "frog" : put "3" into theGame
    repeat with x = 3 to 18
        puppetsprite x, false
    end repeat
```

```
    go to frame "black"
    go to movie "Rhyme8"
```

```
    "farmer" : put "2" into theGame
    repeat with x = 3 to 18
        puppetsprite x, false
    end repeat
```

```
    go to frame "black"
    go to movie "Eggs8"
```

```
end case
```

```
repeat with g = 3 to 8
    puppetsprite g, false
end repeat
```

```
else
```

```
    alert "You must first select a player"
    go to marker (0)
```

```
end if
```

```
end if
```

```

end

on enterkeypass
  global greckeeper, wheredoIGO

  put getuserpassword(greckeeper) into rightpassword

  if line 1 of field "enter pass" = "052096" then
    go to frame "superpass"
    exit
  end if

  if line 1 of field "enter pass" = rightpassword then
    if wheredoIGO = "preferences" then
      set the keyUpScript = ""
      initPrefs
    else
      if wheredoIGO = "dataview" then
        set the keyUpScript = ""
        initDataView
      else
        if wheredoIGO = "player" then
          set the keyUpScript = ""
          initplayer
        else
          if wheredoIGO = "delete" then
            set the keyUpScript = ""
            initdelete
          end if
        end if
      end if
    end if
  end if

  end if

else
  put "" into member "fake pass"
  put "" into member "enter pass"
  updatestage
  set the keyUpScript = ""
  alert "Incorrect Password! Please Re-Enter Password and Try Again."
  -- set the keyUpScript = "if the key = RETURN then enterkeypass"
end if

end

```

```

on getconNames
  put GetUserNames(greckeeper) into field "allNames"
  -- put "allnames = " & allNames
  repeat with x = 1 to 3
    set targetfield = "spot" && x
    put line x of field "allNames" into field targetfield
    set the font of member targetfield = "Helvetica"
    set the fontsize of member targetfield = 14
  end repeat
end

```



```

    set the forecolor of member targetfield = the forecolor of member "spot 1"

end repeat
updatestage
end

on getproNames howmany
    put GetUserNames(grecordkeeper) into field "allNames"
    -- put "allnames = " & allNames

    if voidP(howmany) then
        repeat with x = 1 to 26
            set targetfield = "spot" && x

            if line x of field "allNames" = "" then
                exit repeat
            end if

            put line x of field "allNames" into field targetfield
            set the font of member targetfield = "Helvetica"
            set the fontsize of member targetfield = 14
            set the forecolor of member targetfield = the forecolor of member "spot 1"

        end repeat
    else
        -- beep
        put "FULL redraw"
        put the number of lines of field "allNames" into num
        repeat with x = 1 to (num+2)
            set targetfield = "spot" && x

            put line x of field "allNames" into field targetfield
            set the font of member targetfield = "Helvetica"
            set the fontsize of member targetfield = 14
            set the forecolor of member targetfield = the forecolor of member "spot 1"

        end repeat
    end if

    updatestage
end

on checkname
    global pro
    cursor 4
    put the number of lines of field "enter name" into NumVer
    repeat with x = 1 to NumVer
        if line x of field "enter name" = "" then
            delete line x of field "enter name"
        end if
    end repeat
end

```



```

put the number of chars of field "enter name" into numchars
put "number of chars in name = "& numchars
if numchars >25 then
    alert "Player Names are limited to 25 characters"
    go to frame "enter1"
    exit
end if

put the text of member "enter name" into tryname
put the number of lines of field "allnames" into numnames
if numchars <> 0 then
    repeat with x = 1 to numchars
        if char x of field "enter name" = "," then
            alert "Commas can't be used in Player Names."
            go to frame "enter1"
            cursor -1
            exit
        end if
    end repeat
    repeat with x = 1 to numnames
        if line x of field "allnames" = field "enter name" then
            alert "Player"&& tryname && "already used. Please enter another Name."
            go to frame "enter1"
            cursor -1
            exit
        end if
    end repeat

    set the keydownscript = ""
    go to frame "enter2"
    dontPassEvent
    cursor -1
else
    alert "Please enter Player Name"
    go to frame "enter1"
    cursor -1
    exit
end if

-- if pro = 0 then
--     go to frame "con"
-- else
--     go to frame "pro"
-- end if
cursor -1
end

on keycheckname
    if the key = ENTER then
        set the keydownscript = ""
        set the keyUpScript = ""
        checkname
        exit
    end if

    if the key = RETURN then
        set the keydownscript = ""
        set the keyUpScript = ""
        checkname
        exit
    end if

```

```

end if

end
|--
on verifyname
  global greckeeper,pro
  cursor 4
  put the number of lines of field "enter name" into NumVer
  repeat with x = 1 to NumVer
    if line x of field "enter name" = "" then
      delete line x of field "enter name"
    end if
  end repeat
  --
  put the number of chars of field "enter name" into numchars
  put the text of member "enter name" into tryname
  put the number of lines of field "allnames" into numnames
  if numchars <> 0 then
    repeat with x = 1 to numchars
      if char x of field "enter name" = "," then
        alert "Commas can't be used in Player Names."
        go to frame "enter1"
        exit
      end if
    end repeat
    repeat with x = 1 to numnames
      if line x of field "allnames" = field "enter name" then
        alert "Player"&& tryname && "already used. Please enter another Name."
        go to frame "enter1"
        exit
      end if
    end repeat
    -- go to frame "enter2"
  else
    alert "Please enter Player Name"
    go to frame "enter1"
    exit
  end if
  --

set PlayerName = the text of member "enter name"
if PlayerName <> "" then
  set the keydownscript = ""
  cursor 4
  AddUser greckeeper, PlayerName
  if Pro = 0 then
    go to frame "con"
  else
    getpronames
    put the number of lines of field "allnames" into x
    set x = x-1
    put "number of people = " & x
    if x <= 6 then
      go to frame "pro"
    else
      if x <=12 and x > 6 then
        go to frame "pro2"
      end if
    end if
  end if
end if

```

```

    else
        if x <= 18 and x > 12 then
            go to frame "pro3"
        else
            if x <= 24 and x > 18 then
                go to frame "pro4"
            else
                if x <= 28 and x > 24 then
                    go to frame "pro5"
                end if
            end if
        end if
    end if
    end if
    end if
    end if
    -- go to frame "pro"
    end if

else
    go to frame "enter 1"
end if
end

on keyverifyname
    if the key = ENTER then
        verifyname
    end if

    if the key = RETURN then
        verifyname
    end if
end

on initPrefs
    global gObPrefsMan
    cursor 4
    repeat with x = 1 to 48
        puppetSprite x , false
    end repeat
    puppetSound 0

    go to frame "prefs"
    if the number of words in field "UserName" <> 0 then
        put field "userName" into whichUser
        put whichUser into field "UserNameTitle"
        closefakefield
        getPrefs gObPrefsMan, whichUser
    else
        put " " into field "UserNameTitle"
    end if
end

end

on initDataView
    cursor 4
    repeat with x = 1 to 48
        puppetSprite x , false
    end repeat
    updatestage

```

```
go to movie "dataview"  
end
```

```
on initplayer  
  global passuser  
  cursor 4  
  repeat with x = 1 to 48  
    puppetSprite x , false  
  end repeat  
  updatestage  
  put passuser into field "username"
```

```
  nameplacement  
  closefakefield  
end
```

```
on initdelete  
  global passuser  
  cursor 4  
  repeat with x = 1 to 48  
    puppetSprite x , false  
  end repeat  
  updatestage
```

```
  closefakefield  
  set the keyUpScript  
  go to frame "warning"
```

```
end
```

```
on nameplacement nameToPlace  
  global  
  getpronames  
  put the number of lines of field "allnames" into x  
  set x = x-1  
  put "number of names = " & x  
  set y = 0
```

```
  if voidP(nameToPlace) then --MAK changed 1/23/98  
    set nameToPlace = line 1 of field "UserName"  
  else  
    set traceFlag = true  
  end if
```

```
  repeat with y = 1 to x  
    if line y of field "allnames" = nameToPlace then  
      put y into whatplacement  
    end if  
  end repeat
```

```
-- repeat with y = 1 to x  
--   if line y of field "allnames" = line 1 of field "UserName" then  
--     put y into whatplacement  
--   end if  
-- end repeat
```

```
put "whatplacement = "& whatplacement  
set x = whatplacement
```

```
if x <= 6 then  
  go to frame "pro"
```

```
else
```

```
  if x <=12 and x > 6 then  
    go to frame "pro2"
```

```
  else
```

```
    if x <= 18 and x >12 then  
      go to frame "pro3"
```

```
    else
```

```
      if x <=24 and x > 18 then  
        go to frame "pro4"
```

```
      else
```

```
        if x <= 28 and x > 24 then  
          go to frame "pro5"
```

```
        end if
```

```
      end if
```

```
    end if
```

```
  end if
```

```
end if
```

```
put "we're at frame"&& the frame
```

```
end
```

```
on fakefield
```

```
  puppetsprite 38,true
```

```
  if the machinetype < 255 then
```

```
    set the locV of sprite 38 to 189
```

```
    updatestage
```

```
  else
```

```
    set the locV of sprite 38 to 181
```

```
    updatestage
```

```
  end if
```

```
end
```

```
on closefakefield
```

```
  puppetsprite 38,false
```

```
  put "" into member "fake pass"
```

```
end
```

Score Script5

```
on exitFrame
  global gObPrefsMan
  if soundbusy(1) then
    else
      puppetsound 1, "main track"
      updatestage
    end if
  updateGameIcons gObPrefsMan
  go to the frame
end
```

Script of Cast Member6

```
on mouseUp
  global gRecordKeeper
  set newData = 0
  put the text of member "name" & Return into newData
  put the text of member "age" & Return after newData
  put the text of member "nickname" & Return after newData
  storeData gRecordKeeper, "user1", newData, true

end
```

Script of Cast Member7

```
on mouseUp
  global gRecordKeeper
  restoreData gRecordKeeper, "user1", 2, "name"
  restoreData gRecordKeeper, "user1", 3, "age"
  restoreData gRecordKeeper, "user1", 4, "nickname"

end

--RestoreData me, whichCast, whichLine, whichmember
```

Script of Cast Member11

```
on mouseUp
    go to movie "dataview"
end
```

Script of Cast Member12

```
on mouseUp
    global gtheName
    put field "UserName" into theName
    set gtheName = theName
    go to movie "rappers"
end
```

Script of Cast Member13

```
on mouseUp
    global gtheName
    put field "UserName" into theName
    set gtheName = theName
    go to movie "c.c coal.dir"
end
```

Score Script14

Script of Cast Member15

Score Script16

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "3" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to frame "intro" of movie "Rhyme8"
-- set gscoringlevel = the result
-- set gsavedlevel = gscoringlevel
--end
```

```
on mouseUp
    sound stop 1
    puppetSound 0
    cursor 4
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "3" into theGame
    go to frame "black"
    go to movie "Rhyme8"
```

end

Script of Cast Member17

```
on mouseUp
    global gRecordKeeper
    put field "userName" into theName
    addUser(gRecordKeeper, theName)
end
```

Script of Cast Member18

```
on mouseUp
    global gRecordKeeper
    put field "UserName" into theName
    put field "GameNum" into theGame
    setUpRound(gRecordKeeper, theName, Value(theGame))
end
```

Script of Cast Member20

```
on mouseUp
    global gRecordKeeper
    set Numplays = item 1 of field "Scores"
    set NumRight = item 2 of field "scores"
    addtoScore(gRecordKeeper, Value(Numplays), Value(NumRight))
end
```

Script of Cast Member21

```
on mouseUp
    global gRecordKeeper
    put field "Level" into level
    setScoringLevel(gRecordKeeper, level)
end
```

Script of Cast Member22

```
on mouseUp
    global gRecordKeeper
    put field "Level" into level
    setSavedLevel(gRecordKeeper, level)
end
```

Script of Cast Member23

```
on mouseUp
    global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName, gnextgame
    put field "UserName" into theName
    set gtheName = theName
    put field "GameNum" into theGame
    setUpRound(gRecordKeeper, theName, Value(theGame))
    go to frame "intro" of movie "Rhyme8.dir"
    set gscoringlevel = the result
    set gsavedlevel = gscoringlevel
    set gnextgame = gscoringlevel
end
```

Script of Cast Member24

```
on mouseUp
    global gtheName
    put field "UserName" into theName
    set gtheName = theName
    go to movie "Eggs8.dir"
end
```

Score Script25

```
on mouseUp
    put "Player 1" into field "username"
end
```

Score Script26

```
on mouseUp
    put "Player 2" into field "username"
end
```

Score Script27

```
on mouseUp  
    put "Player 3" into field "username"  
end
```

Score Script28

```
on mouseUp  
    put "Player 4" into field "username"  
end
```

Score Script29

```
on mouseUp  
    put "Player 5" into field "username"  
end
```

Score Script30

```
on mouseUp  
    put "Player 6" into field "username"  
end
```

Score Script31

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "6" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to movie "coal8"
-- set gscoringlevel = the result
-- set gsavedlevel = gscoringlevel
--end
```

```
--
on mouseUp
    sound stop 1
    puppetSound 0
    cursor 4
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "6" into theGame
    go to movie "coal8"
```

end

Score Script32

```
--on mouseUp
--  global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
--  put field "UserName" into theName
--  set gtheName = theName
--  put "4" into theGame
--  setUpRound(gRecordKeeper, theName, Value(theGame))
--  go to movie "rappers8"
--  set gscoringlevel = the result
--  set gsavedlevel = gscoringlevel
--end
```

```
on mouseUp
  sound stop 1
  puppetsound 0
  cursor 4
  global gtheName, theGame
  put field "UserName" into theName
  set gtheName = theName
  put "4" into theGame
  go to movie "rappers8"
end
```

Score Script33

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "2" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to movie "Eggs8"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
--end
```

```
on mouseUp
    sound stop 1
    puppetSound 0
    cursor 4
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "2" into theGame
    go to movie "Eggs8"

end
```

Score Script35

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "1" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to movie "katy8"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
--end
```

```
on mouseUp
    sound stop 1
    puppetsound 0
    cursor 4
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "1" into theGame
    go to movie "Katy8"
```

```
end
```

Score Script36

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "5" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to frame "mac"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
-- go to movie "Karloon8"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
--
--end
```

```
on mouseUp
    sound stop 1
    puppetsound 0
    cursor 4
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "5" into theGame
    go to movie "Karloon8"
```

end

Score Script38

```
on exitFrame
    if not soundbusy(1) then
        puppetsound 1, "main track"
        updatetage
    end if
    cursor -1
    -- repeat with x = 3 to 8
    --     puppetsprite x, true
    --     set the visible of sprite x to true
    -- end repeat

end
```

Score Script39

```
on mousedown
    buttonDownhandler
end
on mouseup
    buttonUPhandler
end
```

Script of Cast Member40

```
on mouseUp
    go to movie "dataview"
end
```

Score Script44

```
on mouseUp
  global passuser, wheredoiGO

  if field "spot 1" <> "" then
    put field "spot 1" into passuser

    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if

  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if

end
```

Score Script45

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 2" <> "" then
    put field "spot 2" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script46

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 3" <> "" then
    put field "spot 3" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script47

```
on exitFrame
  go to the frame
end
```

Score Script48

```
on exitFrame
  put "" into field "enter name"
  set the keydownscript = ""
  getconNames
end
```

Score Script49

```
on mouseUp
  checkname
end
```

Score Script50

```
on mouseUp
  global pro
  set the keydownscript = ""
  if pro = 0 then
    go to frame "con"
  else
    go to frame "Pro"
  end if
end
```

Score Script52

```
on mouseUp
  verifyname
end
```

Score Script53

```
on exitFrame
-- set the font of member "enter name" = "helvetica"
-- set the fontsize of member "enter name" = 14
-- set the forecolor of member "enter name" = the forecolor of member "spot 1"

go to the frame

end
```

Score Script54

```
on exitFrame
repeat with x = 3 to 8
  puppetsprite x, false
end repeat

puppetsprite 17, false
puppetsprite 18, false
put "" into field "enter name"
set the keydownscript = "keycheckname"
cursor -1
end
```

Score Script55

```
on exitFrame
set the keydownscript = "keyverifname"
end
```

Score Script56

```
on mouseUp
  repeat with x = 1 to 48
    puppetsprite x, false
  end repeat

  if the machinetype = 256 then
    go to frame "win"
  else
    go to frame "mac"
  end if

  halt
end
```

Score Script58

```
on mouseUp
  global gthename, thegame
  if field "username" <> "" then
    set thegame = 0
    set gthename = field "username"
    cursor 4
    puppetsprite 17, false
    puppetsprite 18, false
    go to movie "progress"
  else
    alert "Please select a Player"
  end if

end
```

Score Script59

```
on mouseUp
  if field "spot 10" <> "" then
    put field "spot 10" into field "username"
  else
    go to frame "enter1"
  end if
end
```

Score Script60

```
on mouseUp
  if field "spot 11" <> "" then
    put field "spot 11" into field "username"
  else
    go to frame "enter1"
  end if
end
```

Score Script62

```
on mouseUp
  if field "spot 12" <> "" then
    put field "spot 12" into field "username"
  else
    go to frame "enter1"
  end if
end
```

Score Script63

```
on exitFrame
  put "" into field "enter name"
  set the keydownscript = ""
  getproNames
end
```

Score Script64

```
on mouseUp
  global wheredoIGO
  --cursor 4

  repeat with x = 1 to 48
    puppetSprite x , false
  end repeat

  set wheredoIGO = "Dataview"
  go to frame "enterdatapass"
end
```

Score Script65

```
on mousedown
  buttonDownhandler
end
on mouseup
  arrowuphandler
end
```

Score Script66

```
on mousedown  
  buttonDownhandler  
  -- arrowDownhandler  
end  
  
on mouseup  
  arrowuphandler  
end
```

Score Script67

```
on mouseUp  
  global passuser, wheredoiGO  
  if field "spot 4" <> "" then  
    put field "spot 4" into passuser  
    if passuser <> field "userName" then  
      set wheredoiGO = "player"  
      repeat with x = 1 to 48  
        puppetsprite x, false  
      end repeat  
  
      go to frame "enterplayerpass"  
    end if  
  
  else  
    repeat with x = 3 to 8  
      puppetsprite x, false  
    end repeat  
  
    puppetsprite 17, false  
    puppetsprite 18, false  
    go to frame "enter1"  
  end if  
  
end
```

Score Script68

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 5" <> "" then
    put field "spot 5" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script70

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 6" <> "" then
    put field "spot 6" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```


Score Script83

```
on mouseUp
    global passuser, wheredoiGO
    if field "spot 7" <> "" then
        put field "spot 7" into passuser
        if passuser <> field "userName" then
            set wheredoiGO = "player"
            repeat with x = 1 to 48
                puppetsprite x, false
            end repeat
            go to frame "enterplayerpass"
        end if
    else
        repeat with x = 3 to 8
            puppetsprite x, false
        end repeat
        puppetsprite 17, false
        puppetsprite 18, false
        go to frame "enter1"
    end if
end
```

Score Script84

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 8" <> "" then
    put field "spot 8" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script85

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 9" <> "" then
    put field "spot 9" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script88

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 10" <> "" then
    put field "spot 10" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "ente:playerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script89

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 11" <> "" then
    put field "spot 11" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script96

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 12" <> "" then
    put field "spot 12" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script92

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 13" <> "" then
    put field "spot 13" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Script of Cast Member93:UserName

Score Script95

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 14" <> "" then
    put field "spot 14" into passuser
    if passuser <> field "userName" then
      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script96

```
on mouseUp
    global passuser, wheredoiGO
    if field "spot 15" <> "" then
        put field "spot 15" into passuser
        if passuser <> field "userName" then
            set wheredoiGO = "player"
            repeat with x = 1 to 48
                puppetsprite x, false
            end repeat

            go to frame "enterplayerpass"
        end if
    else
        repeat with x = 3 to 8
            puppetsprite x, false
        end repeat

        puppetsprite 17, false
        puppetsprite 18, false
        go to frame "enter1"
    end if
end
```

Score Script97

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 16" <> "" then
    put field "spot 16" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script98

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 17" <> "" then
    put field "spot 17" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script100

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 18" <> "" then
    put field "spot 18" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script127

```
on mouseUp
  global passuser, wheredoiG0
  if field "spot 19" <> "" then
    put field "spot 19" into passuser
    if passuser <> field "userName" then

      set wheredoiG0 = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if

  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script128

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 20" <> "" then
    put field "spot 20" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if

  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script129

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 21" <> "" then
    put field "spot 21" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script130

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 22" <> "" then
    put field "spot 22" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if

  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script132

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 23" <> "" then
    put field "spot 23" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if

  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script133

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 24" <> "" then
    put field "spot 24" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if

  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script134

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 25" <> "" then
    put field "spot 25" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script135

```
on mouseUp
  global passuser, wheredoiGO
  if field "spot 26" <> "" then
    put field "spot 26" into passuser
    if passuser <> field "userName" then

      set wheredoiGO = "player"
      repeat with x = 1 to 48
        puppetsprite x, false
      end repeat

      go to frame "enterplayerpass"
    end if
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script136

```
on exitFrame
  global greckeeper, Pro, gnewbie
  set gnewbie = 1
  put getProStatus(greckeeper) into Pro

  if pro = 2 then
    put getUserpassword(greckeeper) into checkpass
    put "checkpass = " & checkpass
    if checkpass <> "" then
      -- go to frame "prointro"
      -- getpronames
      nameplacement
    else
      go to frame "newpass"
    end if
  else
    if pro = 1 then
      go to frame "prointro"
      getpronames 666
    else
      if pro = 0 then
        go to frame "con"
      end if
    end if
  end if

end
```

Score Script138

```
on exitFrame
end
```

Score Script139

```
on exitFrame
    if not soundbusy(1) then
        puppetsound 1, "main track"
        updatestage
    end if
end
```

Score Script140

```
on exitFrame
    global gObPrefsMan
    TurnOnGameIcons(gObPrefsMan)
    puppetsprite 17,false
    puppetsprite 18,true
    put "" into field "enter name"
    -- set the keydownscript = ""
    -- getproNames
end
```

Score Script143

```
on exitFrame
    global gObPrefsMan
    TurnOnGameIcons(gObPrefsMan)
    puppetsprite 17,true
    puppetsprite 18,false
    put "" into field "enter name"
    -- set the keydownscript = ""
    -- getproNames
end
```

Score Script144

```
on exitFrame
  global gObPrefsMan
  TurnOnGameIcons(gObPrefsMan)
  puppetsprite 17,true
  puppetsprite 18,false
  put "." into field "enter name"
  -- set the keydownscript = ""
  -- getproNames
end
```

Score Script145

```
on mouseUp
  global greckeeper,Pro,gnewbie
  set gnewbie = 1
  put getProStatus(grecker) into Pro
  if pro = 1 then
    go to frame "prointro"
    getpronames
  else
    if pro = 0 then
      go to frame "con"
    end if
  end if
end
```

Score Script147

```
on exitFrame
  if not soundbusy(1) then
    puppetsound 1, "main track"
    updatestage
  end if
end
```

Score Script149

```
on exitFrame
  global gdataFieldSprites,gthename,grecordkeeper,Pro,gnewbie,gwhatversion

  if gnewbie = 1 then
    put getProStatus(grecordkeeper) into Pro
    --
    --   if the runMode = "author" then
    --     set Pro = 2 -- MAK 1/22 Take out!!!!
    --   end if

    if pro = 2 then
      put getUserPassword(grecordkeeper) into userpass
      if userpass <> "" then
        nameplacement
      else
        go to frame "newpass"
      end if
    else
      alert "Please Insert the Earobics Pro Plus Step 1 CD"
      halt
    end if

  else
    put getnamenummer(grecordkeeper) into serialnum
    put serialnum into field "serialnum"
    put getProStatus(grecordkeeper) into Pro
    --
    --   if the runMode = "author" then
    --     set Pro = 2 -- MAK 1/22 Take out!!!!
    --   end if

    case(pro) of
      2: go to frame "intro"
      1: alert "Please Insert the Earobics Pro CD."
        halt
      0: alert "Please Insert the Earobics CD."
        halt
    end case

  end if
end
```

Score Script150

```
on exitFrame
  go to the frame
end
```

```
on keyup
```

```
  if the key = RETURN then
    if line 1 of field "enter new pass" <> "" then
      go to frame "Confirmpass"
    else
      alert "You must enter a Password"
      go to frame "NewPass"
    end if
  end if
```

```
end if
```

```
if the keycode <> 51 then
  if the key <> RETURN then
    -- put "." before char 1 of member "fake pass"
```

```
    put the number of chars of field "enter new pass" into Xnum
```

```
    put the number of chars of field "fake pass" into Ynum
```

```
    if Ynum > Xnum then
      repeat with x = Ynum down to (Xnum+1)
        delete char x of field "fake pass"
      end repeat
    end if
```

```
    if Ynum < Xnum then
      put Xnum-Ynum into Diff
      if Diff > 1 then
        repeat with x = Ynum to Xnum
          put "." before char 1 of member "fake pass"
        end repeat
      else
        put "." before char 1 of member "fake pass"
      end if
    end if
```

```
    updatestage
```

```
  end if
```

```
else
```

```
  put "" into member "enter new pass"
```

```
  put "" into member "fake pass"
```

```
end if
```

```
end
```

Score Script151

```
on exitFrame
  cursor -1
  put "" into member "fake pass"
  put "" into member "enter new pass"
  fakefield
end
```

Score Script152

```
on keydown
  beep
end

on keyup
  beep
end

on exitFrame
  pause
end
```

Score Script153

Score Script154

on exitFrame

end

Score Script155

on exitFrame

end

Score Script156

```
--on exitFrame
-- global pro
-- global grecordkeeper
-- put getProStatus(grecordkeeper) into Pro
-- set pro = 1 --test
--
-- if pro = 2 then
--   repeat with x = 44 to 46
--     puppetsprite x,true
--     set the visible of sprite x to true
--   end repeat
--
--   put getUserpassword(grecordkeeper) into checkpass
--   put "checkpass = " & checkpass
--   if checkpass <> "" then
--     continue
--   else
--     go to frame "newpass"
--   end if
-- else
--   repeat with x = 44 to 46
--     puppetsprite x,true
--     set the visible of sprite x to false
--   end repeat
-- end if
--
--end
```

Score Script167

```
on exitFrame
  go to the frame
end

on keyup
  if the key = RETURN then
    if line 1 of field "enter new pass" = line 1 of field "confirm new pass" then
      setuserpassword(grecordkeeper,line 1 of field "confirm new pass")
      -- go to frame "prointro"
      -- getpronames
      closefakefield
      go to frame "storedpass"
    else
      alert "Password not confirmed, Please try again."
      closefakefield
      go to frame "NewPass"
    end if
  end if

end if

if the keycode <> 51 then
  if the key <> RETURN then
    -- put "." before char 1 of member "fake pass"

    put the number of chars of field "confirm new pass" into Xnum
    put the number of chars of field "fake pass" into Ynum

    if Ynum > Xnum then
      repeat with x = Ynum down to (Xnum+1)
        delete char x of field "fake pass"
      end repeat
    end if

    if Ynum < Xnum then
      put Xnum-Ynum into Diff
      if Diff > 1 then
        repeat with x = Ynum to Xnum
          put "." before char 1 of member "fake pass"
        end repeat
      else
        put "." before char 1 of member "fake pass"
      end if
    end if

    updatestage
  end if
else
  put "." into member "confirm new pass"
  put "." into member "fake pass"
end if
end
```

Score Script168

```
on exitFrame
  put "" into member "fake pass"
  put "" into member "confirm new pass"
  fakefield
end
```

Score Script169

```
on exitFrame
  set the keyUpScript = "if the key = RETURN then enterkeypass"
  go to the frame
end

on keyup
  global wheredoIGO

  --
  -- if the key = RETURN then
  --   enterkeypass
  --
  -- end if

  if the keycode <> 51 then
    if the key <> RETURN then
      -- put "." before char 1 of member "fake pass"

      put the number of chars of field "enter pass" into Xnum
      put the number of chars of field "fake pass" into Ynum

      if Ynum > Xnum then
        repeat with x = Ynum down to (Xnum+1)
          delete char x of field "fake pass"
        end repeat
      end if

      if Ynum < Xnum then
        put Xnum-Ynum into Diff
        if Diff > 1 then
          repeat with x = Ynum to Xnum
            put "." before char 1 of member "fake pass"
          end repeat
        else
          / put "." before char 1 of member "fake pass"
        end if
      end if

      updatetage
    end if
  else
    put "." into member "enter pass"
    put "." into member "fake pass"
  end if
end
```

Score Script170

```
on exitFrame
  put "" into member "fake pass"
  put "" into member "enter pass"
  fakefield
  set the keyUpScript = "if the key = RETURN then enterkeypass"
end
```

Script of Cast Member171

Score Script173

```
on mouseUp
  set the keyUpScript = ""
  closefakefield
  nameplacement
end
```

Score Script174

```
on exitFrame
  go to the frame
end
on keyup
  if the key = RETURN then
    cursor 4
    go to frame "prointro"
    getpronames 666
  end if
end
```

Score Script176

```
on exitFrame
  go to the frame
end

on keyup
  global wheredoigo
  if the key = RETURN then

    if line 1 of field "enter new pass" = line 1 of field "confirm new pass" then
      setUserpassword(grecordkeeper, line 1 of field "confirm new pass")

      if wheredoIGO = "preferences" then
        initPrefs
      else
        if wheredoIGO = "dataview" then
          initDataView
        else
          if wheredoIGO = "player" then
            initplayer
          end if
        end if
      end if
    end if

  else
    alert "Password not confirmed, Please try again."
    closefakefield

    go to frame "superpass"
  end if

end if

if the keycode <> 51 then
  if the key <> RETURN then
    -- put "." before char 1 of member "fake pass"

    put the number of chars of field "confirm new pass" into Xnum
    put the number of chars of field "fake pass" into Ynum

    if Ynum > Xnum then
      repeat with x = Ynum down to (Xnum+1)
        delete char x of field "fake pass"
      end repeat
    end if

    if Ynum < Xnum then
      put Xnum-Ynum into Diff
      if Diff > 1 then
        repeat with x = Ynum to Xnum
          put "." before char 1 of member "fake pass"
        end repeat
      else
        put "." before char 1 of member "fake pass"
      end if
    end if
  end if
end if
```

```
end if
    updatestage
end if
else
    put "" into member "confirm new pass"
    put "" into member "fake pass"
end if
end
```

Score Script184

```
on exitFrame
  set the keyupscri: ""
  go to the frame
end
```

```
on keyup
```

```
  if the key = RETURN then
```

```
    go to frame "superpassconfirm"
  end if
```

```
  if the keycode <> 51 then
```

```
    if the key <> RETURN then
```

```
      -- put "." before char 1 of member "fake pass"
```

```
      put the number of chars of field "enter new pass" into Xnum
```

```
      put the number of chars of field "fake pass" into Ynum
```

```
      if Ynum > Xnum then
```

```
        repeat with x = Ynum down to (Xnum+1)
```

```
          delete char x of field "fake pass"
```

```
        end repeat
```

```
      end if
```

```
      if Ynum < Xnum then
```

```
        put Xnum-Ynum into Diff
```

```
        if Diff > 1 then
```

```
          repeat with x = Ynum to Xnum
```

```
            put "." before char 1 of member "fake pass"
```

```
          end repeat
```

```
        else
```

```
          put "." before char 1 of member "fake pass"
```

```
        end if
```

```
      end if
```

```
      updatestage
```

```
    end if
```

```
  else
```

```
    put "." into member "enter new pass"
```

```
    put "." into member "fake pass"
```

```
  end if
```

```
end
```

Score Script185

```
on mouseUp
  closefakefield
  go to frame "prointro"
  getpronames 666
end
```

Score Script186

```
on exitFrame
  cursor -1
end
```

Score Script187

```
on mouseUp
  closefakefield
  enterkeypass
end
```

Score Script191

```
on mouseUp
  closefakefield
  go to frame "newpass"
end
```

Score Script194

```
on mouseUp
  if line 1 of field "enter new pass" = line 1 of field "confirm new pass" then
    setUserpassword(grecordkeeper,line 1 of field "confirm new pass")

    closefakefield
    go to frame "storedpass"
  else
    alert "Password not confirmed, Please try again."
    closefakefield
    go to frame "NewPass"
  end if
end
```

Score Script196

```
on mouseUp
  if field "spot 2" <> "" then
    put field "spot 2" into field "username"
  else
    repeat with x = 3 to 8
      puppetsprite x,false
    end repeat

    puppetsprite 17,false
    puppetsprite 18,false
    go to frame "enter1"
  end if
end
```

Score Script197

```
on mouseUp
  if field "spot 3" <> "" then
    put field "spot 3" into field "username"
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script199

```
on mouseUp
  if field "spot 1" <> "" then
    put field "spot 1" into field "username"
  else
    repeat with x = 3 to 8
      puppetsprite x, false
    end repeat

    puppetsprite 17, false
    puppetsprite 18, false
    go to frame "enter1"
  end if
end
```

Score Script200

```
on exitFrame  
  go to frame "logos"  
end
```

Score Script201

```
on exitFrame  
  global pro  
  case(pro) of  
    "0": go to frame "parent"  
    "1": go to frame "pro"  
    "2": go to frame "plus"  
  end case  
end
```

Score Script208

```
on mouseUp  
  repeat with x = 1 to 48  
    puppetSprite x , false  
  end repeat  
  go to movie "proview"  
end
```

Score Script209

```
on exitFrame  
  go to the frame  
end
```

Score Script213

```
on exitFrame  
  put empty before line 1 of field "pref Help"  
end
```

Score Script214

```
on mouseUp
    -- need this script to block
    -- passing of mouseUp to sprite in menu frames
    nothing
end
```

Score Script215

```
on mouseUp
    if line 1 of field "enter new pass" <> "" then
        go to frame "Confirmpass"
    else
        alert "You must enter a Password"
        go to frame "NewPass"
    end if
end
```

Score Script216

```
on mouseUp
  global wheredoIGO
  if line 1 of field "enter new pass" = line 1 of field "confirm new pass" then
    setUserpassword(grecordkeeper,line 1 of field "confirm new pass")
    closefakefield
    if wheredoIGO = "preferences" then
      initPrefs
    else
      if wheredoIGO = "dataview" then
        initDataView
      else
        if wheredoIGO = "player" then
          initplayer
        end if
      end if
    end if
  end if

  else
    alert "Password not confirmed, Please try again."
    closefakefield
    go to frame "superpass"
  end if
end
```

Score Script217

```
on mouseUp
  if line 1 of field "enter new pass" <> "" then
    closefakefield
    go to frame "superpassconfirm"
  else
    alert "You must enter a Password"
    closefakefield
    go to frame "superpass"
  end if
end
```

Score Script218

Score Script219

```
on exitFrame
  go to the frame
end
```

Score Script228

```
on mouseUp
    global passuser, wheredoiGO

    if field "userName" <> "" then
        set wheredoiGO = "delete"
        repeat with x = 1 to 48
            puppetsprite x, false
        end repeat

        put field "userName" into passuser
        go to frame "enterdelete"
    end if

end
```

Score Script238

```
on exitFrame
    global passuser
    cursor -1
    put "Are you sure you want to delete player" into line 1 of member "delete text"
    put " " & passuser & "?" after word 8 of line 1 of member "delete text"
end
```

Score Script239

```
on mouseUp
    nameplacement

end
```

Score Script240

```
on mouseUp
  global grecordkeeper,passuser
  cursor 4
  deleteuser(grecordkeeper,the text of field "userName")
  put "" into field "userName"
  getpronames 666
  nameplacement
  cursor -1
end
```

Score Script2

Score Script3

on exitFrame
 go to the frame
end

Score Script4

on mouseUp
 halt
end

Score Script5

on mouseUp
 go to frame "upgrade"
end

Score Script7

on mouseUp
 halt
end

Score Script9

on mouseUp
 donamenu
end

Movie Script10

```
on startmovie
  global greckordkeeper, PlusStatus, OddStatus, version, targetProStatus
  -- set ProStatus greckordkeeper, 2
  set targetProStatus = 2

  set PlusStatus = 1 --set to 0 otherwise
  set OddStatus = 1 -- if we check for odd or even
  set version = "10"
```

```
end
```

```
on stopmovie
  if the runmode = "author" then
    put the text of field "color" into field "name"
    put the text of field "color" into field "number"

  end if
  set the keydownscript = ""
```

```
end
```

```
on checkdonamenumber
  if the key = RETURN then
    donamenumber
    dontPassEvent
  end if
  if the key = ENTER then
    donamenumber
    dontPassEvent
  end if
```

```
end
```

```
on donamenumber
  global greckordkeeper, name, number, PlusStatus, OddStatus, version
  cursor 4
```

```
  put line 1 of field "name" into name
  put line 1 of field "number" into number
  put char 8+PlusStatus of number into dash1
  put char 11+PlusStatus of number into dash2
  put "number = " & number
  put the number of chars of number into totalchars
```

```
  if PlusStatus = 1 then
    if char 1 of number <> "9" then
      alertserial
      exit
    end if
```



```

end if

if dash1 and dash2 <> "-" then
    alertserial
    exit
end if
if totalchars <> 12+plusStatus then
    alertserial
    exit
end if
put char 1 of version into versionchar1
put char 2 of version into versionchar2

if char (9+plusStatus) of number = versionchar1 and char (10+plusStatus) of number = versionchar2 then
    alertserial
    exit
end if

put char 1+plusStatus of number into
put char 2+plusStatus of number into
set x = value(x)
set y = value(y)
set addnum = x+y
set addnum = value(addnum)
put addnum mod(2)
put addnum mod(2) into oddoreven

if oddoreven = OddStatus then
    go to frame "upgrade"

    -- cursor 4
    -- setnamenum greckeeper, name, number:
    -- go to movie "datatest"

else
    alertserial
end if

end

on savenameNum
    global greckeeper, name, number, targetProStatus
    cursor 4
    setProStatus(greckeeper, targetProStatus)
    setnamenum greckeeper, "Cognitive Concepts", targetProStatus
end

on alertserial
    cursor -1
    alert "Incorrect serial number. Please enter it again"
    put "" into field "number"
end

```

```

on findrecords
  global gDataSavePath
  ---open dialog. find application
  cursor 4
  put "" into field "returnmessage"
  put "" into field "returnvalue"
  set driveList to DrivesToList()
  set driveName to getAt(driveList, count(driveList)) -- get last drive name
  if the machineType > 255 then -- PC
    if directoryExists("c:\Earobics") = 0 then
      set driveName to "c:\Earobics"

      if directoryExists("c:\Earobics\Earp4t7") = 0 then
        set driveName to "c:\Earobics\Earp4t7"
      end if
      if directoryExists("c:\Earobics\Earp4t7") = 0 then
        set driveName to "c:\Earobics\Earp4t7"
      end if
      if directoryExists("c:\Earobics\Earp4_7") = 0 then
        set driveName to "c:\Earobics\Earp4_7"
      end if
    end if
  end if

  set driveName to "c:\Earobics"
  end if

  set typeStr to "EXECUTABLE *.EXE" -- "All Files *.* Text Files *.txt"
  set retName to FileOpenDialog(driveName, typeStr, "Open File", True, True)
  if retName = "" then
    set driveName to driveName & " "
    set typeStr to "ARPL TEXT" -- "TEXT/W6BN/MV95" -- text, word 6, director 5
    set retName to FileOpenDialog(driveName, typeStr)
  end if
  set the stagecolor to the stagecolor
  put "" into field "returnvalue"
  put retName into field "returnmessage"
  put "retName =" & retName

  ---Checks to see if file "records.cst" exists
  put the number of chars of retname into TotalChars
  set delimiterNum = 0
  if the machineType > 255 then
    set delimiter = "\"
  else
    set delimiter = ":"
  end if
  --find the drivename
  repeat with x = 1 to totalchars
    if char x of retname = delimiter then
      set driven delimiter = x
      exit repeat
    end if
  end repeat
  put "drivedelimiter =" & driven delimiter
  set checkDrive = ""

```

```

repeat with x = 1 to (drivedelimeter-1)
  put char x of rename after char x of checkdrive
end repeat
put "checkDrive =" & checkDrive
set realdrive = 0
repeat with x in DrivesToList()
  if x = checkDrive then
    set realdrive = 1
  end if
end repeat

if realdrive = 1 then
  repeat with x = totalchars down to 1
    if char x of rename = delimiter then
      set delimiterNum = x
      exit repeat
    end if
  end repeat
  put delimiterNum

  set CurrDir = ""
  repeat with x = 1 to delimiterNum
    put char x of rename after char x of currdir
  end repeat
  put "currdir =" & currdir
  set retfile to FileExists( currdir & "records.cst" )
  put retfile into field "returnvalue"
  put GetMessage(retfile) into field "returnmessage"

  --found file,copy default prefs

  if retfile = 0 then
    --Changes gDataSavePath to new directory

    put gDataSavePath into OGgDataSavePath
    put "OGgDataSavePath =" & OGgDataSavePath
    set gDataSavePath = currdir

    put getprostatus(grecordkeeper) into UpgradePro
    put "UpgradePro =" & UpgradePro
    -- reset gDataSavePath
    put OGgDataSavePath into gDataSavePath
    put "gDataSavePath =" & gDataSavePath
    closerecords grecordkeeper

    if UpgradePro = 1 then
      -- copy files
      set the filename of castLib "temp.cst" = currdir&"records.cst" --sample open
records
      put the number of members of castlib "temp.cst" into totalMem
      openrecords grecordkeeper
      duplicate member 1 of castlib "temp.cst",member 1 of castlib "records.cst"
      repeat with x = 1 to totalMem
        duplicate member x of castlib "temp.cst",member x of castlib "records.cst"
      end repeat
    end repeat
  end if
end repeat

```

```

    set the filename of castLib "temp.cst" = the pathname & "temp.cst"

    saverecords greckeeper
    closerrecords greckeeper
    saveNameNum
    cursor -1
    alert "Import Successful! Your data records have been imported from Earobics PRO
to Earobics PRO PLUS."
    cursor 4
    -- halt
    go to movie "datatest"
    -- end if

else
    closerrecords greckeeper
    cursor -1
    alert "Not a valid Earobics PRO data records file"
end if

else
    closerrecords greckeeper
    cursor -1
    alert "Not a valid Earobics PRO data records file"
end if
else
end if
end if
end
end

```

```

--on delete
-- global gDataSavePath
-- set deleteValue to DeleteFile(gDataSavePath & "records.cst")
-- put deleteValue into field "returnvalue"
-- put GetMessage(deleteValue) into field "returnmessage"
-- put deleteValue
--
--
--end

--on copy
-- global gDataSavePath
-- set copyVal to CopyFile("CCI:test:Earobics PRO Step 1:records.cst",gDataSavePath &
"records.cst")
-- put copyVal into field "returnvalue"
-- put GetMessage(copyVal) ----into field "returnmessage"
-- put "copyVal =" & copyVal
--end

```

```

on GetMessage theNum
case theNum of
0: set message to "successful completion"
-1: set message to "General error of unknown origin"
-5: set message to "File deletion failure"
-6: set message to "File rename failure"

```

```
-7: set message to "File not found"
-8: set message to "Specified file is actually a directory"
-9: set message to "File creation failure"
-10: set message to "File open failure"
-11: set message to "File write failure"
-12: set message to "File close failure"
-13: set message to "File read failure"
-14: set message to "Destination disk full"
-15: set message to "Directory not found"
-16: set message to "Specified directory is actually a file"
-17: set message to "Directory creation failure"
-18: set message to "Could not delete specified directory"
-19: set message to "Could not retrieve directory ID number"
-40: set message to "Could not allocate memory for file copy"
-51: set message to "Specified drive does not exist"
-52: set message to "Specified drive exists but is not mounted"
-61: set message to "Specified drive is not a CD-ROM"
-210: set message to "New filename already exists or two paths are different"
otherwise set message to "unknown error code"
end case
```

Script of Cast Member12

Score Script13

```
on exitFrame
  put "" into field "name"
  put "" into field "number"
  set the keydownscript = "checkdonamenumber"
end
```

Score Script14

```
on exitFrame
  cursor -1
end
```

Score Script15

on exitFrame

```
-- set the forecolor of member "agreement text2" = the forecolor of member "agreement  
color"  
-- set the backColor of member "agreement text2" = the backcolor of member "agreement  
color"  
set the stagecolor to the stagecolor  
end
```

Score Script16

on exitFrame

```
go to the frame  
end
```

Score Script17

on exitFrame

```
set driveList to DrivesToList()  
set driveName to getAt(driveList, count(driveList)) -- get last drive name  
if the machineType > 255 then -- PC  
set driveName to driveName & "\"  
set typeStr to "All Files/.../Text Files/*.txt"  
set retName to FileOpenDialog(driveName, typeStr, "Open File", True, True)  
else -- Mac  
set driveName to driveName & ":"  
set typeStr to "TEXT/W6BN/MV95" -- text, word 6, director 5 documents  
set retName to FileOpenDialog(driveName, typeStr)  
end if  
  
put "" into field "returnvalue"  
put retName into field "returnmessage"  
end
```

Score Script18

```
on exitFrame
    set retVal to CopyFile(field "filename1",field "filename2")
    put retVal into field "returnvalue"
    put GetMessage(retVal) into field "returnmessage"
end
```

Score Script19

```
on mouseUp
    findrecords
end
```

Score Script20

```
on exitFrame
    cursor -1
end
```

Score Script22

```
on mouseUp
    savenameNum
    go to movie "Datatest"
end
```

Parent Script:drum:ner

```
--4/10/97
-- added command to turn on notesSprites to try to fix
-- bug where notes come on then go off

property ancestor, myHandlers, playList, currentLevel, soundNum, UserHits, RoundScoreList
Property audioOn, interval, Game, armMember, timeOutNum, drumsound, disableRePlayButton
Property SkipYesOrNo

on x-----Public Handlers-----
-- I'm a separator
end

on new me
-- pub.
global gGameScorer, gLEDMan
set ancestor = gGameScorer
set myHandlers = 0
set myHandlers = GetMyHandlers(me)
set Game = #drumSounds
set gLEDMan = 0
set gLEDMan = newscript "LEDDisplayManager", 41,42)
-- set up LED display object, params are the sprite nums
return me
end

on SetUpTest me, Level, PrefList
global gRoundOver, gNoter
noteSpritesOn gNoter
set CurrentLevel = level
setUpPlayList me
set soundNum = 0
set TimeOutNum = 0
set drumsound = the memberNum of member "drum"
preloadmember drumsound
set the purgePriority of member drumsound = 0
set RoundScoreList = []
set ScoreList = [0,0,0]
setat scoreList.1, currentLevel
setat RoundScoreList.1, scorelist
if currentLevel > 3 then
    set audioOn = false
    set armMember = the Membernum of member "arm Wave"
else
    set audioOn = true
    set armMember = the Membernum of member "arm down"
end if
PreLoadMember armMember
set the purgepriority of member armMember = 1
case (CurrentLevel) of
    1,4: set Interval = 60
    2,5: set Interval = 30
    3,6: set Interval = 15
end case
set gRoundOver = false
set disableRePlayButton = getAt(PrefList,1)
set SkipYesOrNo = getAt(PrefList,2)
end
```



```

on Playsounds me
  global gRoundOver, gSoundsPlaying, gLEDMan
  if count(playList) < 1 then exit
  set gSoundsPlaying = true
  set soundNum = getat(PlayList,1)
  set UserHits = 0
  deleteat PlayList,1
  repeat with x = 1 to soundNum
    puppetsound member drumsound
    updatestage
    repeat while soundBusy (1)
      nothing
    end repeat
    if x = soundNum then exit repeat
    wait interval
  end repeat
  StartUserDrumTime 300, "CheckUserHits gGame"
  startLEDDisplay gLEDMan, 300 -- start LED timer display
  if count(playList) < 1 then set gRoundOver = true
  puppetsound 0
  set gSoundsPlaying = false
  -- put soundNum
end

on RepeatSounds me
  global gLEDMan
  -- ClearLEDDisplay gLEDMan
  repeat with x = 1 to soundNum
    puppetsound member drumsound
    updatestage
    repeat while soundBusy (1)
      nothing
    end repeat
    if x = soundNum then exit repeat
    wait interval
  end repeat
  StartUserDrumTime (300, "CheckUserHits gGame")
  startLEDDisplay gLEDMan, 300 -- start LED timer display
  puppetsound 0
end

on userDrumHit me
  global gLEDman
  global gArmSprite
  set timeOutNum = 0
  set UserHits = userHits + 1
  set the member of sprite gArmsprite = member armMember
  updatestage
  putUpNote UserHits
  wait 4
  -- put userHits
  if userHits > soundNum then
    doWrongScore me, #over
  else
    if AudioOn then
      puppetsound member drumsound
      updatestage
      set the member of sprite gArmsprite = member "Arm up"
    end if
  end if
end

```

```

    updatestage
    repeat while soundBusy(1)
        UpDateLEDDisplay gLEDman
    end repeat
end if
set the member of sprite gArmsprite = member "Arm up"
updatestage
puppetsound 0
end if

end

```

```

on xx-----Private Handlers-----
    -- i'm a separator
end

```

```

on unLoadSounds me
    -- this almost inert script needs to be here to be
    -- in compliance with the three other game scripts
    -- The command is issued from the score and
    -- doesn't really matter to this game
    -- however the game will crash without it.
    set the purgePriority of member drumsound = 1
end

```

```

on setUpPlayList me
    -- priv.
    -- sets Up list of ten values from 1 to 4
    -- randomly distributed but with no more than
    -- two same number beats in a row
    set PlayList = []
    set BeatNumList = {1,2,3,4}
    set BeatNums = Count(beatNumList)
    repeat while count(playList) < 10
        set x = count(PlayList)
        set beats = getat(beatNumList, random(BeatNums))
        if x > 1 then
            set B1 = getat(PlayList, x-1)
            set B2 = getat(playList, x)
            if beats = b1 and beats = b2 then next repeat
        end if
        append playlist, beats
    end repeat
    put playList
end

```

```

on xxx-----Testing Handlers-----
    -- i'm a separator
    nothing
end

```

```

on showHandlers me
    -- Testing
    -- puts list of handlers in message window
    put myHandlers
end

on showProps me
    -- testing
    -- puts list of properties and their current values in message window
    set PropNum = count(me)
    repeat with x = 1 to PropNum
        set prop = 0
        set thisProp = getpropat(me, x)
        if thisProp = #myHandlers then next repeat
        put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop
        put prop
    end repeat
end

```

Parent Script2:IsoSound

```
--4/10/97
-- added command to turn on notesSprites to try to fix
--- bug where notes come on then go off
-- changed purge Priority in preloadSounds handler from 1 to 0

property ancestor, myHandlers, playList, currentLevel, SoundNum, UserHits, RoundScoreList
Property audioOn, interval, shortVowelList, LongVowelList, consonantList, ListToPlay,
game
property armMember, TimeOutNum, badWordList, disableRePlayButton, SkipYesOrNo

on x-----Public Handlers -----
  -- I'm a separator
end

on new me
  -- pub.
  global gGameScorer, gLEDMan
  set ancestor = gGameScorer
  set myHandlers = 0
  set myHandlers = GetMyHandlers(me)
  setUpSoundLists me
  setUpBadWordList me
  set game = #IsoSounds
  set gLEDMan = 0
  set gLEDMan = new(script "LEDDisplayManager", 41,42)
  -- set up LED display object, params are the sprite nums
  return me
end

on SetUpTest me, Level, PrefList
  global gRoundOver, gNoter
  notesSpritesOn gNoter
  set CurrentLevel = level
  setUpPlayList me
  set SoundNum = 0
  set TimeOutNum = 0
  set ListToPlay = []
  set RoundScoreList = []
  set ScoreList = [0,0,0]
  setat scoreList, 1, currentLevel
  setat RoundScoreList 1, scorelist
  if currentLevel > 9 then
    set audioOn = false
    set armMember = the Membernum of member "arm wave"
  else
    set audioOn = true
    set armMember = the Membernum of member "arm down"
  end if
  PreLoadMember armMember
  set the purgepriority of member armMember = 1
  case (CurrentLevel) of
    7,10: set InterVal = 60
    8,11: set InterVal = 30
    9,12: set Interval = 15
  end case
  set gRoundOver = false
```

```

    set disableRePlayButton = getAt(PrefList,1)
    set SkipYesOrNo = getAt(PrefList,2)
end

on PlaySounds me
    global gRoundOver, gSoundsPlaying, gLEDMan
    if count(playList) < 1 then exit
    set gSoundsPlaying = true
    set ListToPlay = getAt(PlayList,1)
    preloadsounds (me, listtoplay)
    set UserHits = 0
    deleteat PlayList,1
    set SoundNum = count(listToPlay)
    repeat with x = 1 to SoundNum
        set sound = getAt(listToPlay, x)
        puppetsound sound
        updatestage
        repeat while soundBusy(1)
            nothing
        end repeat
        if x = SoundNum then exit repeat
        wait interval
    end repeat
    StartUserDrumTime 300, "CheckUserHits gGame"
    startLEDDisplay gLEDMan, 300 -- start LED timer display
    if count(playList) < 1 then set gRoundOver = true
    repeat while soundBusy(1)
        nothing
    end repeat
    set gSoundsPlaying = false
    puppetsound 0
end

on RepeatSounds Me
    global gLEDMan
    -- ClearLEDDisplay gLEDMan
    set SoundNum = count(listToPlay)
    repeat with x = 1 to SoundNum
        set sound = getAt(listToPlay, x)
        puppetsound sound
        updatestage
        repeat while soundBusy(1)
            nothing
        end repeat
        if x = SoundNum then exit repeat
        wait interval
    end repeat
    StartUserDrumTime 300, "CheckUserHits gGame"
    startLEDDisplay gLEDMan, 300 -- start LED timer display
    puppetsound 0
end

on userDrumHit me
    global gArmSprite, gLEDman
    set timeOutNum = 0
    set UserHits = userHits + 1
    set the member of sprite gArmSprite = member armMember
    updatestage
    putUpNote UserHits
    wait 4

```

```

if userHits > SoundNum then
  doWrongScore me, #over
else
  if AudioOn then
    set sound = getat(listToPlay, userHits)
    puppetsound sound
    updatestage
    set the member of sprite gArmsprite = member "Arm up"
    updatestage
    repeat while soundBusy(1)
      UpDateLEDDisplay gLEDman
    end repeat
    puppetsound 0
  end if
end if
set the member of sprite gArmsprite = member "Arm up"
updatestage
end

```

```

on xx-----Private Handlers-----
  -- i'm a separator
end

```

```

on preloadSounds me, ListOfSounds
  repeat with thesound in listOfSounds
    preLoadMember theSound
    put the result
    set the purgePriority of member theSound to 0
  end repeat
end

```

```

on unloadSounds me
  if voidP(listToPlay) or listToPlay = [] then exit
  repeat with thesound in listToPlay
    unloadMember theSound
    set the purgePriority of member theSound to 3
  end repeat
end

```

```

on setUpSoundLists me
  put "looking for missing sounds from IsoSound game"
  set ShortVowelList = []
  set x = the number of lines of field "shortVowels"
  repeat with y = 1 to x
    append shortVowelList, line y of field "shortVowels"
    if the number of member line y of field "shortVowels" = -1 then
      put line y of field "shortVowels" && "is missing"
    next repeat
  end if
  append shortVowelList, line y of field "shortVowels"
end repeat
set LongVowelList = []
set x = the number of lines of field "LongVowels"
repeat with y = 1 to x
  append LongVowelList, line y of field "LongVowels"
  if the number of member line y of field "LongVowels" = -1 then

```

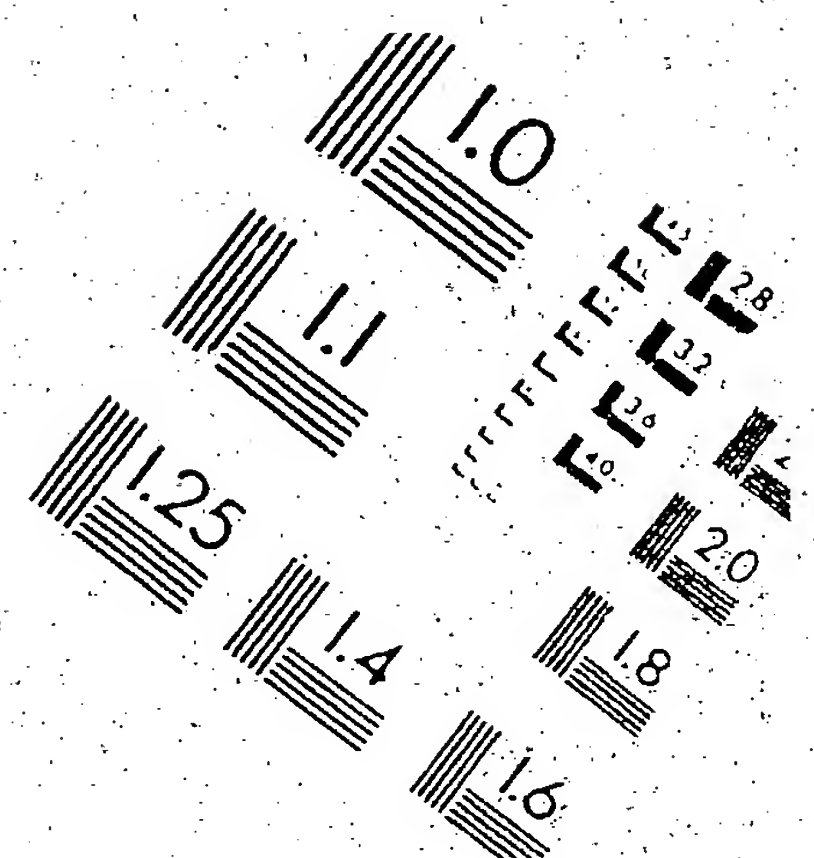
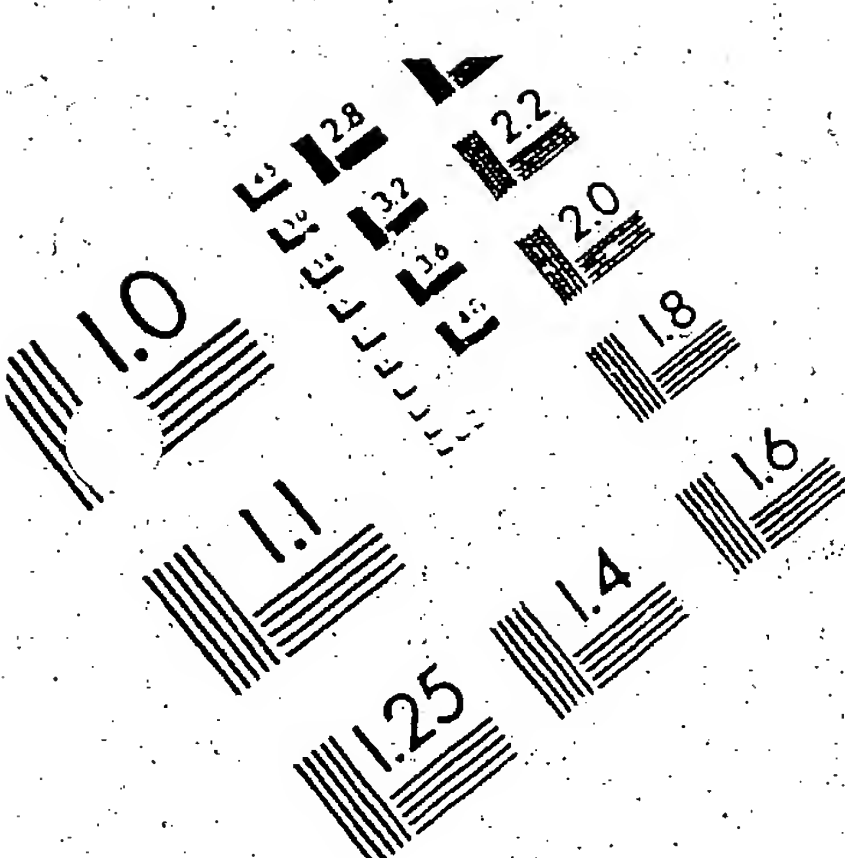
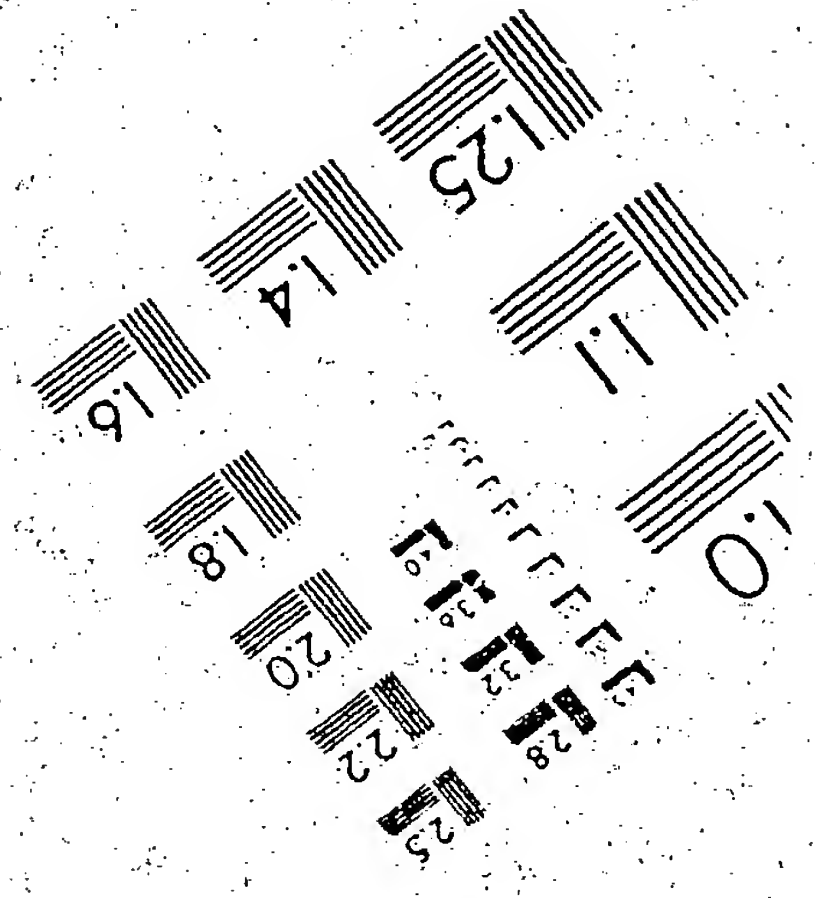
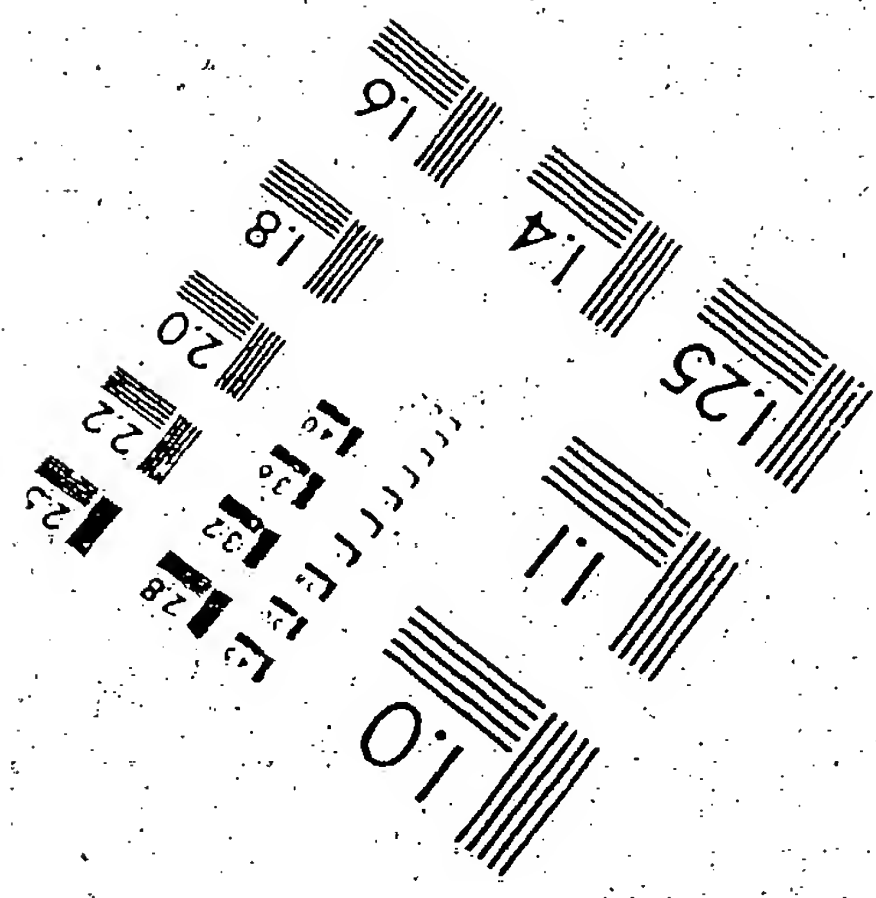
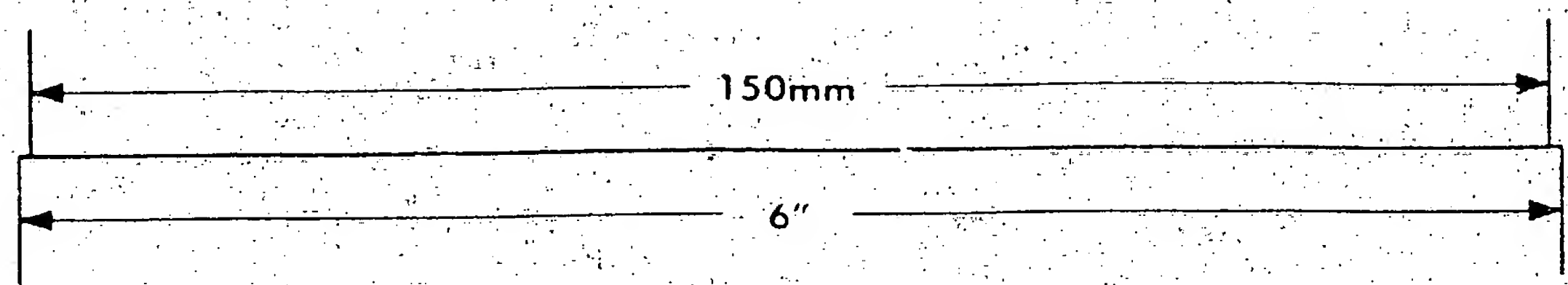
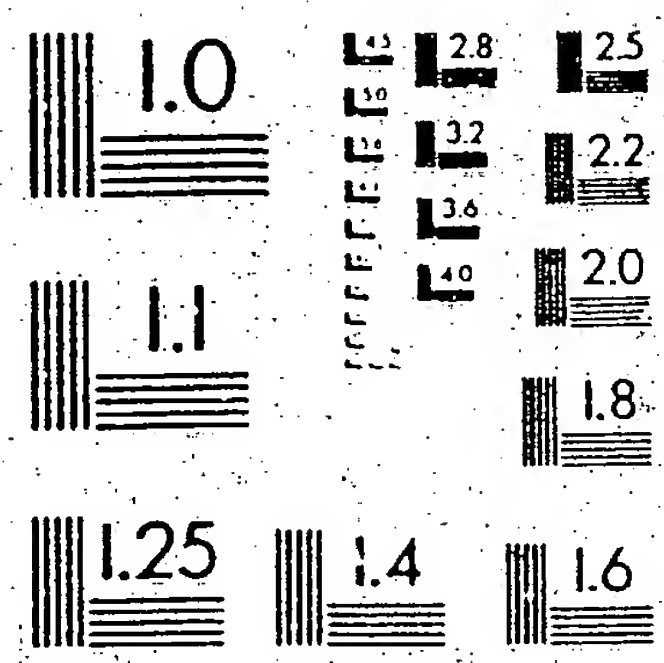



IMAGE EVALUATION
TEST TARGET QA-3



APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone 716-482-0300
Fax 716-288-5989

```

    put line y of field "LongVowels" && "is missing"
  next repeat
end if
append LongVowelList, line y of field "LongVowels"
end repeat
set consonantList = []
set x = the number of lines of field "consonants"
repeat with y = 1 to x
  append consonantList, line y of field "consonants"
  if the number of member line y of field "consonants" = -1 then
    put line y of field "consonants" && "is missing"
  next repeat
end if
append consonantList, line y of field "consonants"
end repeat
put "done looking"
end

on setUpPlayList me
  -- priv.
  -- sets up playlist with no sound repeating
  -- more than twice in a row
  set PlayList = []
  set BeatNumList = [1,2,3,4]
  set BeatNums = Count(beatNumList)
  repeat while count(playList) < 10
    set x = count(PlayList)
    set beats = getat(beatNumList, random(BeatNums))
    if x > 1 then
      set B1 = getat(PlayList, x-1)
      set B2 = getat(playList, x)
      if beats = b1 and beats = b2 then next repeat
    end if
    append playList, beats
  end repeat

  repeat with x = 1 to 10
    set TempList = []
    set NumSounds = (getat(pLayList, x))
    set tryAgain = false
    repeat while tryAGain = true or not ListP(getat(pLayList, x))
      -- if tempList that results from next repeat is a tabu list
      -- or if we haven't converted this position of playlist yet
      -- from beats (integer) to a list then we keep trying
      repeat with y = 1 to numSounds
        set soundList = random(4)
        case (soundList) of
          1: set ListCount = count(LongVowelList)
             append tempList, getat(longVowelList, random(listCount))
          2: set ListCount = count(shortVowelList)
             append tempList, getat(shortVowelList, random(listCount))
          3,4: set ListCount = count(consonantList)
              append tempList, getat(consonantList, random(listCount))
        end case
      end repeat
    end repeat
    if checkForBadWords(me, tempList) then
      put tempList
      set tryAgain = true
    else
      setat playList, x, tempList
      set tryAgain = false
    end if
  end repeat
end

```



```

        end if
    end repeat
end repeat
put playList
put count(playList)
end

```

```

on checkForBadWords me, listToCheck
-- function checks list of random sounds
-- generated by playlist handler against list
-- of list of naughty words.
-- returns 1 if bad word is found
set isBad = 0
set x = count(badwordList)
repeat with y = 1 to x
    if getat(badwordList, y) = listToCheck then
        set isBad = 1
        return isBad
    end if
end repeat
return isbad
end

```

```

on setUpBadWordList me
-- sets property badwordList by referring to
-- cast member "badWords" which contains
-- line by line list of tabu combinations
set BadWordList = []
set numLines = the number of lines of field "badWords"
repeat with x = 1 to numLines
    if numLines = "" then next repeat
    set tempList = []
    set numWords = the number of words in line x of field "badWords"
    repeat with y = 1 to numWords
        append tempList, word y of line x of field "badWords"
    end repeat
    append badwordList, tempList
end repeat
end

```

```

on oldsetUpPlayList me
-- priv.
-- sets up playlist with no sound repeating
-- more than twice in a row
set PlayList = []
set BeatNumList = [1,2,3,4]
set BeatNums = Count(beatNumList)
repeat while count(playlist) < 10
    set x = count(PlayList)
    set beats = getat(beatNumList, random(BeatNums))
    if x > 1 then
        set B1 = getat(PlayList, x-1)
        set B2 = getat(playList, x)
        if beats = b1 and beats = b2 then next repeat
    end if
    append playlist, beats
end repeat

```

```

end repeat
repeat with x = 1 to 10
  set TempList = {}
  set NumSounds = (getat(pLayList,x))
  set soundList = random(3)
  case (soundList) of
    1:set ListCount = count(LongVowelList)
      repeat with y = 1 to numSounds
        append tempList, getat(longVowelList, random(listCount))
      end repeat
    2:set ListCount = count(shortVowelList)
      repeat with y = 1 to numSounds
        append tempList, getat(shortVowelList, random(listCount))
      end repeat
    3:set ListCount = count(consonantList)
      repeat with y = 1 to numSounds
        append tempList, getat(consonantList, random(listCount))
      end repeat
  end case
  setat playList, x, tempList
end repeat
put playList
end

```

```

on xxx-----Testing Handlers-----
  -- i'm a separator
  nothing
end

```

```

on showHandlers me
  -- Testing
  -- puts list of handlers in message window
  put myHandlers
end

```

```

on showProps me
  -- testing
  -- puts list of properties and their current values in message window
  set PropNum = count(me)
  repeat with x = 1 to PropNum
    set prop = 0
    set thisProp = getpropat(me, x)
    if thisProp = #myHandlers then next repeat
    put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop
    put prop
  end repeat
end

```

Parent Script3: syllables

```
--4/10/97
-- added command to turn on notesSprites to try to fix
-- bug where notes come on then go off
-- changed purge Priority in preloadSounds handler from 1 to 0

property ancestor, myHandlers, playList, currentLevel, SoundNum, UserHits, RoundScoreList
Property audioOn, interval, OneSylWordList, TwoSylWordList, ThreeSylWordList,
FourSylWordList
Property wordToPlay, game, partList, wordparts
property armMember, TimeOutNum, disableRePlayButton, SkipYesOrNo

on x-----Public Handlers -----
  -- I'm a separator
end

on new me
  -- pub.
  global gGameScorer, gLEDMan
  set ancestor = gGameScorer
  set myHandlers = 0
  set myHandlers = GetMyHandlers(me)
  set game = #syllables
  set gLEDMan = 0
  set gLEDMan = new(script "LEDDisplayManager", 41, 42)
  -- set up LED display object, params are the sprite nums.
  return me
end

on SetUpTest me, Level, PrefList
  global gRoundOver, gNoter
  noteSpritesOn gNoter
  set CurrentLevel = level
  setUpWordLists me
  setUpPlayList me
  set SoundNum = 0
  set TimeOutNum = 0
  set wordToPlay = 0
  set wordParts = []
  set RoundScoreList = []
  set ScoreList = [0, 0, 0]
  setat scoreList, 1, currentLevel
  setat RoundScoreList 1, scorelist
  if currentLevel = 14 then
    set audioOn = false
    set armMember = the Membernum of member "arm wave"
  else
    set audioOn = true
    set armMember = the Membernum of member "arm down"
  end if
  PreLoadMember armMember
  set the purgepriority of member armMember = 1
  set interval = 30 --??????
  set gRoundOver = false
  set disableRePlayButton = getAt(PrefList, 1)
  set SkipYesOrNo = getAt(PrefList, 2)
end
```

```

on PlaySounds me
  global gRoundOver, gSoundsPlaying, gLEDMan
  if count(playList) < 1 then exit
  set gSoundsPlaying = true
  set WordToPlay = getat(PlayList,1)
  set WordParts = getat(partList,1)
  preLoadSounds me
  set SoundNum = count(wordParts)
  set UserHits = 0
  deleteat PlayList,1
  deleteat partList,1

  puppetsound member wordToPlay
  put the name of member wordToPlay
  updatestage
  repeat while soundBusy(1)
    nothing
  end repeat
  StartUserDrumTime 300, "CheckUserHits gGame"
  startLEDDisplay gLEDMan, 300 -- start LED timer display
  if count(playList) < 1 then set gRoundOver = true
  set gSoundsPlaying = false
  puppetsound 0
end

on RepeatSounds me
  global gLEDMan
  -- ClearLEDDisplay gLEDMan
  puppetsound member wordToPlay
  put the name of member wordToPlay
  updatestage
  repeat while soundBusy(1)
    nothing
  end repeat
  StartUserDrumTime 300, "CheckUserHits gGame"
  startLEDDisplay gLEDMan, 300 -- start LED timer display
  puppetsound 0
end

on userDrumHit me
  global gLEDman
  global gArmSprite
  set timeOutNum = 0
  set UserHits = userHits + 1
  set the member of sprite gArmSprite = member armMember
  updatestage
  putUpNote UserHits
  wait 4
  if userHits > SoundNum then
    doWrongScore me, #over
  else
    if AudioOn then
      set sound = getat(wordParts, userHits)
      puppetsound member sound
      put the name of member sound
      updatestage
      set the member of sprite gArmSprite = member "Arm up"
      updatestage
      repeat while soundBusy(1)
        UpDateLEDDisplay gLEDman
      end repeat
    end if
  end if
end

```

```

        end repeat
    end if
    set the member of sprite gArmsprite = member "Arm up"
    updatestage
    puppetsound 0
end if

```

```

end

```

```

on xx-----Private Handlers-----
    -- i'm a separator
end

```

```

on preLoadSounds me
    if voidP(wordToPlay) or wordToPlay = 0 then exit
    preloadMember WordtoPlay
    put the result
    set the purgePriority of member WordtoPlay = 0
    repeat with theSound in wordParts
        preloadMember theSound
        put the result
        set the purgePriority of member theSound = 0
    end repeat
end

```

```

on UnLoadSounds me
    if voidP(wordToPlay) or wordToPlay = 0 then exit
    unloadMember WordtoPlay
    set the purgePriority of member WordtoPlay = 3
    repeat with theSound in wordParts
        unloadMember theSound
        set the purgePriority of member theSound = 3
    end repeat
end

```

```

on setUpwordLists me
    set oneSylwordList = []
    set x = the number of lines of field "1SylWords"
    repeat with y = 1 to x
        append oneSylwordList,line y of field "1SylWords"
    end repeat
    set twoSylwordList = []
    set x = the number of lines of field "2SylWords"
    repeat with y = 1 to x
        append twoSylwordList,line y of field "2SylWords"
    end repeat
    set threeSylwordList = []
    set x = the number of lines of field "3SylWords"
    repeat with y = 1 to x
        append threeSylwordList,line y of field "3SylWords"
    end repeat
    set fourSylwordList = []
    set x = the number of lines of field "4SylWords"
    repeat with y = 1 to x
        append fourSylwordList,line y of field "4SylWords"
    end repeat

```



```

end repeat
end

```

```

on setUpPlayList me

```

```

-- priv.

```

```

-- sets up playlist with no sound repeating

```

```

-- more than twice in a row and no more than 3 of each sound permitted

```

```

set PlayList = []

```

```

set PartList = []

```

```

set BeatNumList = [1,2,3,4]

```

```

set BeatNums = Count(beatNumList)

```

```

set ones = 0

```

```

set twos = 0

```

```

set threes = 0

```

```

set fours = 0

```

```

repeat while count(playList) < 10

```

```

    set x = count(PlayList)

```

```

    set beats = getAt(beatNumList, random(BeatNums))

```

```

    if x > 1 then

```

```

        set B1 = getAt(PlayList, x-1)

```

```

        set B2 = getAt(playList, x)

```

```

        if beats = b1 and beats = b2 then next repeat

```

```

    end if

```

```

case (beats) of

```

```

    1 : set ones = ones + 1

```

```

    if ones > 3 then next repeat

```

```

    2: set twos = twos + 1

```

```

    if twos > 3 then next repeat

```

```

    3: set threes = threes + 1

```

```

    if threes > 3 then next repeat

```

```

    4: set fours = fours + 1

```

```

    if fours > 3 then next repeat

```

```

end case

```

```

append playList, beats

```

```

end repeat

```

```

put playList

```

```

repeat with x = 1 to 10

```

```

    set numSyls = getAt(playList, x)

```

```

    case (numSyls) of

```

```

        1: set listPos = random(count(OneSylWordList))

```

```

        set theWord = getAt(OneSylWordList, listPos)

```

```

        deleteAt OneSylWordList, listPos

```

```

        set TheSound = the number of member theword of castlib "lsylwrds.cst"

```

```

        set TheParts = []

```

```

        append theParts the number of member theword of castlib "lsylwrds.cst"

```

```

        2: set listPos = random(count(twoSylWordList))

```

```

        set theWord = getAt(twoSylWordList, listPos)

```

```

        deleteAt twoSylWordList, listPos

```

```

        set TheSound = the number of member theword of castlib "mSylwrds.cst"

```

```

        set TheParts = []

```

```

        repeat with z = 1 to 2

```

```

            set Part = word z of theword

```

```

            append theParts, the number of member part of castlib "mSylwrds.cst"

```

```

        end repeat

```

```

3: set listPos = random(count(threeSylWordList))
   set theWord = getat(ThreeSylWordList, listPos)
   deleteAt ThreeSylWordList, listPos
   set TheSound = the number of member theword of castlib "mSylwrds.cst"
   set TheParts = []
   repeat with z = 1 to 3
     set Part = word z of theword
     append theParts, the number of member part of castlib "mSylwrds.cst"
   end repeat

4: set listPos = random(count(fourSylWordList))
   set theWord = getat(FourSylWordList, listPos)
   deleteAt FourSylWordList, listPos
   set TheSound = the number of member theword of castlib "mSylwrds.cst"
   set TheParts = []
   repeat with z = 1 to 4
     set Part = word z of theword
     append theParts, the number of member part of castlib "mSylwrds.cst"
   end repeat

end case
setat playList, x, theSound
append partList, theParts
end repeat
end

```

```

on xxx-----Testing Handlers-----
  -- i'm a separator
  nothing
end

```

```

on showHandlers me
  -- Testing
  -- puts list of handlers in message window
  put myHandlers
end

```

```

on showProps me
  -- testing
  -- puts list of properties and their current values in message window
  set PropNum = count(me)
  repeat with x = 1 to PropNum
    set prop = 0
    set thisProp = getpropat(me, x)
    if thisProp = #myHandlers then next repeat
    put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop
    put prop
  end repeat
end

```

```

on TestWordList me, whichField, startwhere
  -- plays all words in a "sound name" field and attempts to play all
  -- the word parts of that field. if it brings up and
  -- error alert you no something is wrong with that word or
  -- its word parts
  if whichField = "lsylwords" then
    set castLibName = "lsylwrds.cst"
  end if
end

```

```

else
  set castLibname = "mSylwrds.cst"
end if
if voidP(startwhere) then
  set y = 1
else
  set y = startwhere
end if
set numlines = the number of lines of field whichField
repeat with x = y to numlines
  set thesound = line x of field whichField
  set wordNum = the number of words in thesound
  put x &&"the sound" && "memberNum" && the memberNum of member thesound of
castLib castLibname
  set Sound = the number of member thesound of castLib castLibname
  puppetsound member sound
  updatestage
  repeat while soundbusy(1)
    nothing
  end repeat
  repeat with z = 1 to wordNum
    set soundpart = word z of thesound
    put soundpart
    set soundpart = the number of member soundpart of castlib castlibname
    puppetsound member soundpart
    updatestage
    repeat while soundbusy(1)
      nothing
    end repeat
  end repeat
end repeat
end

```

on FindMissingSoundsandSegments me

```

-- tests all sounds from the sylword lists
-- and looks for all parts of those sounds
-- if parts are missing, posts info to the message window
put "looking for missing sounds from Syllables game"
setUpwordLists me
repeat with theWord in oneSylwordList
  if the number of member theword = -1 then
    put theword && "is missing"
  end if
  repeat with y = 1 to 1
    set thePart = word y of theword
    if the number of member thepart = -1 then
      put thePart && "of"&& theWord&&"is missing"
    end if
  end repeat
end repeat

```

```

repeat with theWord in twoSylwordList
  if the number of member theword = -1 then
    put theword && "is missing"
  end if
  repeat with y = 1 to 2
    set thePart = word y of theword
    if the number of member thepart = -1 then
      put thePart && "of"&& theWord&&"is missing"
    end if
  end repeat
end repeat

```



```

    end if
  end repeat
end repeat

repeat with theWord in threeSylwordList
  if the number of member theword = -1 then
    put theword && "is missing"
  end if
  repeat with y = 1 to 3
    set thePart = word y of theword
    if the number of member thepart = -1 then
      put thePart && "of"&& theWord&&"is missing"
    end if
  end repeat
end repeat

repeat with theWord in fourSylwordList
  if the number of member theword = -1 then
    put theword && "is missing"
  end if
  repeat with y = 1 to 4
    set thePart = word y of theword
    if the number of member thepart = -1 then
      put thePart && "of"&& theWord&&"is missing"
    end if
  end repeat
end repeat
put "done looking"
return 1
end

```

Parent Script4:soundSegments

```
--4/10/97
-- added command to turn on notesSprites to try to fix
-- bug where notes come on then go off
-- changed purge Priority in preloadSounds handler from 1 to 0

property ancestor, myHandlers, playList, currentLevel, SoundNum, UserHits, RoundScoreList
Property audioOn, interval, TwoSegWordList, ThreeSegWordList, FourSegWordList
Property wordToPlay, game, partList, wordParts
property armMember, TimeOutNum, disableRePlayButton, SkipYesOrNo

on x-----Public Handlers -----
  -- I'm a separator
end

on new me
  -- pub.
  global gGameScorer, gLEDMan
  set ancestor = gGameScorer
  set myHandlers = 0
  set myHandlers = GetMyHandlers(me)
  set game = #soundSegments
  set gLEDMan = 0
  set gLEDMan = new(script "LEDDisplayManager", 41, 42)
  -- set up LED display object, params are the sprite nums
  return me
end

on SetUpTest me, Level, PrefList
  global gRoundOver, gNoter
  notesSpritesOn gNoter
  set CurrentLevel = level
  setUpWordLists me
  setUpPlayList me
  set SoundNum = 0
  set TimeOutNum = 0
  set wordToPlay = 0
  set wordParts = 0
  set RoundScoreList = []
  set ScoreList = [0, 0, 0]
  setat scoreList, 1, currentLevel
  setat RoundScoreList 1, scorelist
  if currentLevel = 16 then
    set audioOn = false
    set armMember = the Membernum of member "arm wave"
  else
    set audioOn = true
    set armMember = the Membernum of member "arm down"
  end if
  PreLoadMember armMember
  set the purgepriority of member armMember = 1
  set interval = 30 --??????
  set gRoundOver = false
  set disableRePlayButton = getAt(PrefList, 1)
  set SkipYesOrNo = getAt(PrefList, 2)
end

on PlaySounds me
```

```

global gRoundOver, gSoundsPlaying, gLEDMan
if count(playList) < 1 then exit
set gSoundsPlaying = true
set WordToPlay = getat(PlayList,1)
set wordParts = getat(PartList,1)
preloadSounds me
set UserHits = 0
deleteat PlayList,1
deleteat PartList,1
set SoundNum = count(wordParts)
puppetsound member wordToPlay
updatestage
put the name of member wordToPlay
repeat while soundBusy(1)
    nothing
end repeat
StartUserDrumTime 300, "CheckUserHits gGame"
startLEDDisplay gLEDMan, 300 -- start LED timer display
if count(playList) < 1 then set gRoundOver = true
set gSoundsPlaying = false
puppetsound 0
end

on RepeatSounds me
    global gLEDMan
    -- ClearLEDDisplay gLEDMan
    puppetsound member wordToPlay
    updatestage
    put the name of member wordToPlay
    repeat while soundBusy(1)
        nothing
    end repeat
    StartUserDrumTime 300, "CheckUserHits gGame"
    startLEDDisplay gLEDMan, 300 -- start LED timer display
    puppetsound 0
end

on userDrumHit me
    global gArmSprite, gLEDMan
    set timeOutNum = 0
    set UserHits = userHits + 1
    set the member of sprite gArmSprite = member armMember
    updatestage
    putUpNote UserHits
    wait 1
    if userHits > SoundNum then
        doWrongScore me, #over
    else
        if AudioOn then
            set sound = getat(wordParts, userHits)
            puppetsound member sound
            updatestage
            set the member of sprite gArmSprite = member "Arm up"
            updatestage
            put the name of member sound
            repeat while soundbusy(1)
                updateLEDDisplay gLEDMan
            end repeat
        end if
        set the member of sprite gArmSprite = member "Arm up"
    end if
end

```

```

    updatestage
    puppetsound 0
end if

end

on xx-----Private Handlers-----
    -- i'm a separator
end

on preLoadSounds me
    if voidP(wordToPlay) or wordToPlay = 0 then exit
    preloadMember WordtoPlay
    put the result
    set the purgePriority of member WordtoPlay = 0
    repeat with theSound in wordParts
        preloadMember theSound
        put the result
        set the purgePriority of member theSound = 0
    end repeat
end

on UnLoadSounds me
    if voidP(wordToPlay) or wordToPlay = 0 then exit
    unloadMember WordtoPlay
    set the purgePriority of member WordtoPlay = 3
    repeat with theSound in wordParts
        unloadMember theSound
        set the purgePriority of member theSound = 3
    end repeat
end

on setUpwordLists me
    set twoSegwordList = []
    set x = the number of lines of field "2SegWords"
    repeat with y = 1 to x
        append twoSegwordList,line y of field "2SegWords"
    end repeat
    set threeSegwordList = []
    set x = the number of lines of field "3SegWords"
    repeat with y = 1 to x
        append threeSegwordList,line y of field "3SegWords"
    end repeat
    set fourSegwordList = []
    set x = the number of lines of field "4SegWords"
    repeat with y = 1 to x
        append fourSegwordList,line y of field "4SegWords"
    end repeat
end

on setUpPlayList me

```

```

-- priv.
-- sets up playlist with no sound repeating
-- more than twice in a row
set Playlist = []
set partlist = []
set BeatNumList = {2,3,4}
set BeatNums = Count(beatNumList)
repeat while count(playlist) < 10
    set x = count(Playlist)
    set beats = getat(beatNumList, random(BeatNums))
    if x > 1 then
        set B1 = getat(Playlist, x-1)
        set B2 = getat(playlist, x)
        if beats = b1 and beats = b2 then next repeat
    end if
    append playlist, beats
end repeat
put playlist
set x = 1
repeat while x < 11
    set missingSound = 0 -- flag for missing cast members
    set numSegs = getat(playlist, x)
    case (numSegs) of
        2: set listPos = random(count(twoSegWordList))
            set theWord = getat(twoSegWordList, listPos)
            deleteAt twoSegWordList, listPos
            set TheSound = the number of member theword of castlib "sndSegs.cst"
            if TheSound = -1 then
                set MissingSound = 1
            end if
            set TheParts = []
            repeat with z = 1 to 2
                set Part = word z of theword
                append theParts, the number of member part of castlib "sndSegs.cst"
                if the number of member part of castlib "sndSegs.cst" = -1 then
                    set MissingSound = 1
                end if
            end repeat
        3: set listPos = random(count(threeSegWordList))
            set theWord = getat(threeSegWordList, listPos)
            deleteAt threeSegWordList, listPos
            set TheSound = the number of member theword of castlib "sndSegs.cst"
            if TheSound = -1 then
                set MissingSound = 1
            end if
            set TheParts = []
            repeat with z = 1 to 3
                set Part = word z of theword
                append theParts, the number of member part of castlib "sndSegs.cst"
                if the number of member part of castlib "sndSegs.cst" = -1 then
                    set MissingSound = 1
                end if
            end repeat
        4: set listPos = random(count(fourSegWordList))
            set theWord = getat(fourSegWordList, listPos)
            deleteAt fourSegWordList, listPos
            set TheSound = the number of member theword of castlib "sndSegs.cst"
            if TheSound = -1 then
                set MissingSound = 1
            end if
    end case
    set x = x + 1
end repeat

```

```

end if
set TheParts = []
repeat with z = 1 to 4
    set Part = word z of theword
    append theParts; the number of member part of castlib "sndSegs.cst"
    if the number of member part of castlib "sndSegs.cst" = -1 then
        set MissingSound = 1
    end if
end repeat
end case
if missingSound > 0 then -- some cast member was missing so don't use word
    next repeat
else
    setat playlist, x, thesound
    append partList, theparts
    set x = x + 1
end if
end repeat
end

```

```

--on oldsetUpPlayList me
-- -- priv.
-- -- sets up playlist with no sound repeating
-- -- more than twice in a row
-- set Playlist = []
-- set partlist = []
-- set BeatNumList = [2,3,4]
-- set BeatNums = Count(beatNumList)
-- repeat while count(playlist) < 10
--     set x = count(Playlist)
--     set beats = getat(beatNumList, random(BeatNums))
--     if x > 1 then
--         set B1 = getat(Playlist, x-1)
--         set B2 = getat(playlist, x)
--         if beats = b1 and beats = b2 then next repeat
--     end if
--     append playlist, beats
-- end repeat
-- put playlist
-- repeat with x = 1 to 10
--     set numSegs = getat(playlist, x)
--     case (numSegs) of
--         2: set listPos = random(count(twoSegWordList))
--            set theWord = getat(twoSegWordList, listPos)
--            deleteAt twoSegWordList, listPos
--            set TheSound = the number of member theword of castlib "sndSegs.cst"
--            set TheParts = []
--            repeat with z = 1 to 2
--                set Part = word z of theword
--                append theParts, the number of member part of castlib "sndSegs.cst"
--            end repeat
--     end of
-- end repeat

```



```

--      3: set listPos = random(count(threeSegWordList))
--      set theWord = getat(threeSegWordList, listPos)
--      deleteAt threeSegWordList, listPos
--      set TheSound = the number of member theword of castlib "sndSegs.cst"
--      set TheParts = []
--      repeat with z = 1 to 3
--          set Part = word z of theword
--          append theParts, the number of member part of castlib "sndSegs.cst"
--      end repeat
--
--      4: set listPos = random(count(fourSegWordList))
--      set theWord = getat(fourSegWordList, listPos)
--      deleteAt fourSegWordList, listPos
--      set TheSound = the number of member theword of castlib "sndSegs.cst"
--      set TheParts = []
--      repeat with z = 1 to 4
--          set Part = word z of theword
--          append theParts, the number of member part of castlib "sndSegs.cst"
--      end repeat
--
--      end case
--      setat playList, x, thesound
--      append partList, theparts
--  end repeat
--end

```

```

on xxx-----Testing Handlers-----

```

```

-- i'm a separator
nothing
end

```

```

on showHandlers me
-- Testing
-- puts list of handlers in message window
put myHandlers
end

```

```

on showProps me
-- testing
-- puts list of properties and their current values in message window
set PropNum = count(me)
repeat with x = 1 to PropNum
    set prop = 0
    set thisProp = getpropat(me, x)
    if thisProp = #myHandlers then next repeat
    put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop
    put prop
end repeat
end

```

```

on TestWordList me, whichField, startwhere
-- plays all words in a sound name field and attempts to play all
-- the word parts of that field

```

```

set castLibname = "sndSegs.cst"

```

```

if voidP(startwhere) then
    set y = 1
else
    set y = startwhere

```

```

end if
set numlines = the number of lines of field whichField
repeat with x = y to numlines
  set thesound = line x of field whichField
  set wordNum = the number of words in thesound
  put x &&"thesound" &&"memberNum" && the memberNum of member thesound of
castLib castLibname
  set Sound = the number of member thesound of castLib castLibname
  puppetsound member sound
  updatestage
  repeat while soundbusy(1)
    nothing
  end repeat
  wait 6
  repeat with z = 1 to wordNum
    set soundpart = word z of thesound
    put soundpart
    set soundpart = the number of member soundpart of castlib castlibname
    puppetsound member soundpart
    updatestage
    repeat while soundbusy(1)
      nothing
    end repeat
    wait 6
  end repeat
end repeat
end

```

on FindMissingSoundsandSegments me

```

-- tests all sounds from the soundSg lists
-- and looks for all parts of those sounds
-- if parts are missing ,posts info to the message window
setUpwordLists me
put "looking for missing sounds from SoundSegments game"
repeat with theWord in twoSegwordList
  if the number of member theword = -1 then
    put theword && "is missing"
  end if
  repeat with y = 1 to 2
    set thePart = word y of theword
    if the number of member thepart = -1 then
      put thePart && "of"&& theWord&&"is missing"
    end if
  end repeat
end repeat

```

```

repeat with theWord in threeSegwordList
  if the number of member theword = -1 then
    put theword && "is missing"
  end if
  repeat with y = 1 to 3
    set thePart = word y of theword
    if the number of member thepart = -1 then
      put thePart && "of"&& theWord&&"is missing"
    end if
  end repeat
end repeat

```

```

repeat with theWord in fourSegwordList

```



```

if the number of member theword = -1 then
  put theword && "is missing"
end if
repeat with y = 1 to 4
  set thePart = word y of theword
  if the number of member thepart = -1 then
    put thePart && "of" && theWord && "is missing"
  end if
end repeat
end repeat
put "gone looking"
end

```

Parent Script5:GameScorer

--4/9/97

property ancestor, myHandlers

on new me

global gReccrdKeeper

set ancestor = gRecordKeeper

set myHandlers = 0

set myHandlers = GetMyHandlers(me)

return me

end

on reportScores me

set scoreList = getat(the RoundScoreList of me,1)

set NumPlays = getat (scoreList,2)

if the TimeOutNum of me = 1 then

-- -- got here on a timeout so delete one play for last try

-- set numPlays = numplays - 1

-- setat scorelist, 2, numplays

end if

set NumRight = getat (scoreList,3)

if float(numRight)/float(numPlays) >= .80 and numplays = 10 then

set LeveltoSave = the currentLevel of me + 1

else

set leveltosave = the currentLevel of me

end if

if leveltosave > 17 then set leveltosave = 17

-- put "RoundScoreList =" && the RoundScoreList of me

-- put "leveltosave =" && levelToSave

saveRoundScores (me,the RoundScoreList of me,leveltosave)

end

on resetPlay me

global gNoteSprites

-- to run after a replay of sounds

set the userHits of me = 0

repeat with thisSprite in gNoteSprites

set the loc of sprite thisSprite = point(1000,1000)

end repeat

updatestage

end

on CheckUserHits me

if the userHits of me = 0 then set the TimeOutNum of me = the TimeOutNum of me + 1

if the TimeOutNum of me = 2 then

doTimeOutAbort me

exit

end if

if the userHits of me = the soundNum of me then

doRightScore me

else

doWrongScore me, #under

end if

end

```

on doRightScore me
  global gWhichNote
  set scoreList = getat(the roundscoreList of me,1) -- do scoring
  set NumPlays = getat (scoreList,2)
  set NumPlays = NumPlays + 1
  set NumRight = getat (scoreList,3)
  set NumRight = NumRight + 1
  setat(scoreList,2,NumPlays)
  setat(scoreList,3,NumRight)
  setat(the roundscorelist of me,1,scorelist)
  set gWhichNote = "right Note"
  set x = Random(4)
  if disableRePlayButton() = false then
    -- if we are showing replay button we need
    -- to get rid of it
    set the loc of sprite 4 = point(-1000,-1000)
    set the loc of sprite 5 = point(-1000,-1000)
    updatestage
    puppetSprite 4, false
    puppetSprite 5, false
  end if
  go to frame "Right" && x
end

```

```

on doWrongScore me, WrongHow
  global gWhichNote
  set gWhichNote = "wrong Note"
  if disableRePlayButton() = false then
    -- if we are showing replay button we need
    -- to get rid of it
    set the loc of sprite 4 = point(-1000,-1000)
    set the loc of sprite 5 = point(-1000,-1000)
    updatestage
    puppetSprite 4, false
    puppetSprite 5, false
  end if
  go to frame "wrong"
  if wrongHow = #over then
    set whichNote = the userHits of me
    doOverDisplay whichNote
  end if
  set scoreList = getat(the roundscoreList of me,1) -- do scoring
  set NumPlays = getat (scoreList,2)
  set NumPlays = NumPlays + 1
  setat(scoreList,2,NumPlays)
  setat(the roundscorelist of me,1,scorelist)
end

```

```

on doTimeoutAbort me
  -- if user doesn't click mouse for
  -- two soundplays we abort the round
  -- if there are more than 1 NumPlays then
  -- user did play some in the round so
  -- we report the scores after correcting for the
  -- first timeout.
  global gArmSprite
  set scoreList = getat(the roundscoreList of me,1) -- do scoring
  set NumPlays = getat (scoreList,2)

```

```

if NumPlays < 2 then
  if disableRePlayButton() = false then
    -- if we are showing replay button we need
    -- to get rid of it
    set the loc of sprite 4 = point(-1000,-1000)
    set the loc of sprite 5 = point(-1000,-1000)
    updatestage
    puppetSprite 4, false
    puppetSprite 5, false
  end if
  go to Label ( "QuitorGoOn(end)" ) + 1 -- skip to prompt directly, bypass pref test
  exit
end if
set NumPlays = NumPlays + 1
setat(scoreList,2,NumPlays)
setat(the roundscorelist of me,1,scorelist)
spriteListOff
puppetsprite gArmsprite, false
reportscores me
if disableRePlayButton() = false then
  -- if we are showing replay button we need
  -- to get rid of it
  set the loc of sprite 4 = point(-1000,-1000)
  set the loc of sprite 5 = point(-1000,-1000)
  updatestage
  puppetSprite 4, false
  puppetSprite 5, false
end if
go to Label ( "QuitorGoOn(end)" ) + 1 -- skip to prompt directly, bypass pref test
end

on showHandlers me
  -- Testing
  -- puts list of handlers in message window
  put myHandlers
end

on showProps me
  -- testing
  -- puts list of properties and their current values in message window
  set PropNum = count(me)
  repeat with x = 1 to PropNum
    set prop = 0
    set thisProp = getpropat(me, x)
    if thisProp = #myHandlers then next repeat
    put (string (getpropat(me, x))) &&"=" && getaProp(me,thisProp) into prop
    put prop
  end repeat
end

```

Parent Script6:LEDDisplayManager

```
Property pBaseSprite, pLEDSprite, pTimeSpan, pNumLightsUp, pInterval, pLEDLoc, start

on new me, baseSpriteNum, LEDSpriteNum
    -- first param is the timer artwork itself, second param is
    -- the little green bar which climbs up to indicate time passing
    set pBaseSprite = baseSpriteNum
    set pLEDSprite = LEDSpriteNum
    return me
end

on startLEDDisplay me, timeSpan
    set pTimeSpan = timeSpan
    set the member of sprite pLEDSprite = member "green rect"
    put the ticks into start
    set pNumLightsUp = 0
    set pInterval = timeSpan/9 -- number of LED bars
end

on UpdateLEDDisplay me
    global gpausestatus
    if gpausestatus = "resume" then exit
    if the ticks > start + (pInterval - 1) then -- change display, force a little faster
        if pNumLightsUp = 9 then exit -- we're done
        put the ticks into start
        set the trails of sprite pLEDSprite = true
        if pNumLightsUp = 0 then -- first time
            set pLEDLoc = initLEDLoc(me)
            set the loc of sprite pLEDSprite = pLEDLoc
        else
            set pLEDLoc = pLEDLoc - point(0, 7)
            set the loc of sprite pLEDSprite = pLEDLoc
        end if
        updatestage
        set pNumLightsUp = pNumLightsUp + 1
        set the trails of sprite pLEDSprite = false
        set the loc of sprite pLEDSprite = point (-1000,-1000)
    end if
end

on finishLEDDisplay me
    if pNumLightsUp = 9 then exit -- we're done
    repeat with x = (pNumLightsUp + 1) to 9
        set the trails of sprite pLEDSprite = true
        set pLEDLoc = pLEDLoc - point(0, 7)
        set the loc of sprite pLEDSprite = pLEDLoc
        updatestage
        wait 2
    end repeat
end

on ClearLEDDisplay me
    set the loc of sprite pLEDSprite = the loc of sprite pBaseSprite
    set the member of sprite pLEDSprite = member "rapTapTimer"
    updatestage
    set the loc of sprite pLEDSprite = point(-10000,-10000)
    set the member of sprite pLEDSprite = member "Green Rect"
end
```

```

on wipeDownLEDDisplay me
  set the member of sprite pLEDSprite = member "Black rect"
  repeat with x = 1 to pNumLightsUp
    set the trails of sprite pLEDSprite = true
    set the loc of sprite pLEDSprite = pLEDLoc
    set pLEDLoc = pLEDLoc + point(0, 7)
    updatestage
    wait 4
  end repeat
  set the trails of sprite pLEDSprite = false
  set the loc of sprite pLEDSprite = point (-1000,-1000)
end

on initLEDLoc me
  -- need to do this once and while the pBaseSprite is on stage
  return point((the left of sprite pBaseSprite)+ 24, (the bottom of sprite pBaseSprite)
-22)
end

```

Parent Script7:Noter

property NoteSprites

```
on new me
  init me
  return me
```

end

```
on PutUpNote me
  global gWhichNote
  put "NoteSprites=" & NoteSprites
  if count(NoteSprites) < 1 then exit
  set thisNote = getat(NoteSprites, 1)
  deleteat(NoteSprites, 1)
  set the member of sprite thisNote = member gWhichNote
  updatestage
end
```

```
on NotespritesOn me
  set noteList = [39,38,37,36,35,34,33,32,31,30]
  repeat with x in noteList
    puppetsprite x, true
  end repeat
end
```

```
on NotespritesOff me
  set noteList = [39,38,37,36,35,34,33,32,31,30]
  repeat with x in noteList
    set the member of sprite x = member "no Note"
    puppetsprite x, false
    updatestage
    wait 2
  end repeat
end
```

end

```
on Init me
  set NoteSprites = [30,31,32,33,34,35,36,37,38,39]
end
```

Movie Script8

```
on disableRePlayButton
  -- called during play to determine whether or not to
  -- display replay sound icon
  global gGame
  return the disableRePlayButton of gGame
end

on SkipYesOrNo
  -- called during play to determine whether or not to
  -- go to play again prompt screen
  global gGame
  return the SkipYesOrNo of gGame
end
```

Script of Cast Member9

```
on mouseUp
  global gGame
  repeatSounds gGame
end
```

Score Script10:for frame "test" + 1

```
on enterFrame
  if not disableRePlayButton() then
    puppetSprite 4, true -- replayButton sprite
    puppetSprite 5, true -- replayButton invisible square sprite
    set the loc of sprite 4 = point (612, 125)
    set the loc of sprite 5 = point (556, 81)
    updatestage
  end if
end

on exitFrame
  global gArmSprite, ghandcursor
  cursor ghandcursor
  puppetSprite gArmSprite, true -- drummer's arms
  set the mouseDownscript to empty
  checkUserDrumTime
  go to the frame
end

on idle
  global gLEDman
  UpDateLEDDisplay gLEDman
end
```

Score Script11

```
on exitFrame
  if soundbusy(2) then
    go to the frame
  else
    continue
  end if
end
```

Score Script12

```
on exitFrame
  set rightlist = list("metal", "blues", "funk", "pop rock", "soft rock")
  set x = random(5)
  put getat(rightlist, x) into playmusic
  puppetsound 1, playmusic
end
```

Score Script13

```
on exitFrame
  if soundbusy(1) then
  else
    go to frame "endmusic"
  end if
end
```

Score Script14

```
on exitFrame
  checkUserDrumTime
  go to the frame
end

on mouseDown
  global gGame
  userDrumHit gGame
end
```

Score Script15

```
on exitFrame
  sound fadeOut 1, 60
end
```

Score Script16

```
on exitFrame  
  go to the frame  
end
```

Score Script20

```
on exitFrame  
  -- uncomment this script for real play  
  -- and delete part underneath  
  if soundbusy(2) then  
    go to the frame  
  else  
    go to frame "rightmusic"  
  end if  
end
```

Score Script22

Score Script23

```
on exitFrame  
  
end
```

Score Script28

```
on mousedown  
  dontpassevent  
end
```

Movie Script39:StartMovie

```
-- 4/10/97
-- added ResetAllPurgePriorities handler to ensure against
-- leaving some cast members set to priority 0 from the PreLoadSounds
-- handler in the gGame objects. Placed command at "quitorGoOn" frame
-- and at menu button exit.

on startmovie
  global gNoteSprites, gArmSprite, gNoter,
  gGameScorer, gpausestatus, gTheName, goldscoringlevel
  global gSpritesOnList, ghandcursor, goldscoringlist

  set ghandcursor = list()
  append ghandcursor, the number of member "HandCur"
  append ghandcursor, the number of member "HandCurMask"
  set the purgePriority of member "HandCur" = 0
  set the purgePriority of member "HandCurMask" = 0
  cursor 4
  -- cursor ghandcursor

  set gSpritesOnList = {} -- empty out in case values come in from "dataView"
  set gNoteSprites = {10,11,12,13,14}
  set gArmSprite = 3 -- drummer's arm
  PreLoadMember "arm Wave"
  set the purgepriority of member "arm Wave" = 1
  set gNoter = new(script "noter")
  set gGameScorer = new(script "gameScorer")

  set the visible of sprite 45 to false
  set the visible of sprite 46 to false
  set the visible of sprite 47 to false

  set the visible of sprite 9 to false
  set gpausestatus = "pause"

  if objectP(gRecordKeeper) then
    put setUpRound (gRecordKeeper, gTheName, 4) into goldscoringlevel
    put getGameLevelLists(gRecordKeeper, gTheName) into templevellist
    put getat(templevellist,4) into goldscoringlist
  end if

  if the runMode = "author" then
    -- this checks for missing sounds in the cast
    -- by running through all the fields that hold
    -- the sound names. It posts missing words into the message
    -- window. This should probably be whacked out
    -- before the final burn.
    doMissingSoundsCheck
  end if

  -- go to frame 5
end

on StopMovie
  global gGame, GVoid, gRoundOver, gArmSprite, gWhichNote, gNoter
  global gGameScorer, gSpritesOnList
```

```

global gLEDman
-- set gGame = gVoid
-- set gGameScorer = gVoid
set gNoteSprites = gVoid
set gRoundOver = gVoid
set gSoundsPlaying = gVoid
set gArmSprite = gVoid
set gWhichNote = gVoid
set gNoter = gVoid
set gLEDman = gVoid
set gSpritesOnList = gVoid
set the purgePriority of member "HandCur" = 3
set the purgePriority of member "HandCurMask" = 3
end

```

Score Script40

```

on exitFrame
  global gGame
  unloadSounds gGame
  resetAllPurgePriorities -- just in case
  puppetsound 2, "Play again"
end

```

Score Script41

```

on mouseUp
  global gArmSprite
  puppetsprite gArmSprite, false
  puppetsound (2, 0)
  puppetsound (1, 0)
  unload
  go to frame "intro"
end

```

Score Script42

```

on exitFrame
  Global gNoter
  putUpNote gNoter
  puppetsound 2, 0
  puppetsound 2, "way cool"
  updatestage
end

```

Score Script43

```
on exitFrame
  global gNoter
  notespritesOn gNoter
  go to frame "playSounds"
end
```

Score Script44

```
on exitFrame
  global gArmSprite
  puppetSprite gArmSprite, true -- drummer's arms
end
```

Score Script46

```
on mouseUp
  go to frame "black"
  go to movie "progress"
end
```

Score Script47

```
on mouseUp
  global gArmSprite
  puppetsprite gArmsprite, false
  go to frame "intro"
end
```

Score Script48

```
on exitFrame
  repeat with x = 45 to 47
    set the visible of sprite x to false
  end repeat

  preloadmember member "you've got rythm"
  puppetsound 2, "you've got rythm"
end
```

Movie Script55:DrumTimers

```
on StartUserDrumTime howLong, doWhat
  -- start timing while user should
  -- be repeating beats etc.
  Global gUserTimeOutTime, gTimeOutHandler
  put the Ticks + howLong into gUserTimeOutTime
  put doWhat into gTimeOutHandler
end

on CheckUserDrumTime
  -- Put in exitFrame of looping test frame
  -- to check for end of user's repeating time
  global gUserTimeOutTime, gTimeOutHandler, gLEDMan
  if the ticks < gUserTimeOutTime then
    UpDateLEDDisplay gLEDMan
    exit
  else
    finishLEDDisplay gLEDMan
    do gTimeOutHandler
    set gTimeOutHandler = empty
  end if
end

on CancelUserDrumTime
  -- use to cancel out timer of user's drum time.
  global gUserTimeOutTime, gTimeOutHandler
  set gTimeOutHandler = empty
  set gUserTimeOutTime = empty
end
```

Movie Script56:Put Up Notes

```
on PutUpNote whichNote
  global gNoteSprites
  set Notesprite = getat(gNoteSprites,whichNote)
  case (noteSprite) of
    10: set the loc of sprite 10 to point(265,45)
    11: set the loc of sprite 11 to point(298,45)
    12: set the loc of sprite 12 to point(335,45)
    13: set the loc of sprite 13 to point(367,45)
    14: set the loc of sprite 14 to point(397,45)
  end case
  updatestage
end
```

Movie Script57

```
on doOverDisplay userHits
  global gNoteSprites
  set howlong = 5
  set Note = getat(gNotesPrites,UserHits)
  set NoteBreakUpList = ["note 2","note 3"]
  repeat with NoteName in NoteBreakUpList
    set the member of sprite note = member noteName
    updatestage
    wait howlong
  end repeat
  set the loc of sprite note to point(10000,10000)
  updatestage
  set the member of sprite note to member "note 1"
end
```

Score Script58

```
on exitFrame
  global gRoundOver, gIsoSound
  if groundOver = true then
    reportScores gIsoSound
    go to frame "setUpTest"
  else
    wait 60
    go to frame "IsoSoundTest"
  end if
end
```

Movie Script59

```
on LoadMemberNames whichCastLib, first, Last, fieldName, NumWords
  -- Utility to take a consecutive group of members and feed their
  -- names into a field member for future use
  put empty into member fieldName
  set Linenum = 1
  repeat with x = first to last
    if the type of member x of castLib whichCastLib = #empty then next repeat
    if the number of words in the name of member x of castLib whichCastLib = Numwords
    then
      put the name of member x of castLib whichCastLib into line Linenum of member
      fieldName
      set Linenum = Linenum + 1
    end if
  end repeat
end
```

Score Script63

```
on exitFrame
  global gSoundsPlaying, gNoteSprites, ghandcursor
  spritesOnList gNoteSprites
  if gSoundsPlaying = true then
    go to the frame
  else
    go to frame "test"
    cursor ghandcursor
  end if
end
```

Score Script64

```
on exitFrame
  global gGame
  unloadSounds gGame
  wait 45
  cursor 200
end
```

Score Script65

```
on exitFrame
    global gGame,gTheName, gRecordKeeper, gNoter,level,ghandcursor
    cursor ghandcursor
    noteSpritesoff gNoter
    set gGame = 0
    put setUpRound(gRecordKeeper, gTheName,4) into level
    if level = 17 then
        set level = 16
    end if
    put getGamePrefs (gRecordKeeper, gTheName, 4) into PrefList
    put "prefList = "&PrefList

    -- if the runMode = "author" then
    --     set Level = 3 --- take out!!!!!!!!!!
    -- end if
    if level < 7 then
        set gGame = new(script "Drummer")
        setUpTest (gGame, level,PrefList)

        exit
    end if
    if 6 < level and Level < 13 then
        set gGame = new (Script "isoSound")
        setUpTest (gGame, level,PrefList)
        exit
    end if
    if level = 13 or level = 14 then
        set gGame = new (Script "syllables")
        setUpTest (gGame, level,PrefList)
        exit
    end if
    if level = 16 or level = 15 then
        set gGame = new (Script "soundSegments")
        setUpTest (gGame, level,PrefList)
        exit
    end if
end
```

Score Script66

```
on exitFrame
  global gGame, gRoundOver, gArmSprite

  if soundBusy(1) then go to the frame
  if gRoundOver = False then
    -- puppetsprite gArmSprite, false
    set the member of sprite gArmSprite = member "arm up"
    spriteListOff
    puppetSound (1,0)
    puppetSound (2,0)

    -- go to frame "PlaySounds"
    go to frame "listen"
  else
    reportScore gGame
    -- puppetsprite gArmSprite, false
    set the member of sprite gArmSprite = member "arm up"
    spriteListOff
    puppetSound (1,0)
    puppetSound (2,0)

    go to frame "QuitOrGoOn"
  end if
end
```

Score Script67

```
on exitFrame
  go to the frame
end
```

Score Script68

```
on exitFrame
  playSounds gGame
end
```

Script of Cast Member69

```
on mouseUp  
    go to frame "intro"  
end
```

Script of Cast Member70

```
on mouseUp  
    go to movie "dataTest"  
end
```

Score Script 2

```
on exitFrame  
    Global gNoter  
    putUpNote gNoter  
end
```

Score Script75

```
on exitFrame  
    Global gNoter  
    putUpNote gNoter  
    puppetsound 2, "you've got the beat"  
    updatestage  
end
```

Score Script76

```
on exitFrame  
    Global gNoter  
    putUpNote gNoter  
end
```

Score Script77

```
on exitFrame
  Global gNoter
  putUpNote gNoter
  puppetSound 2, "that's right"
  updateStage
end
```

Score Script85

```
on exitFrame
  global gIsoSound, gNoteSprites
  set gNoteSprites = [10,11,12,13,14]
  playSoundSegs gIsoSound
end
```



```

on spritesOnList Spritelist
--Takes a LIST of non-consecutive (or consecutive) channels,
-- and puppets them. Must be passed as a list []
--turns global list gSpritesOnList off first and
-- then turns on spritelist and makes SpriteList
-- into gSpritesOnList
global gSpritesOnList
if VoidP(gSpritesOnList) then set gSpritesOnList = []
if count(gSpritesOnList) > 0 then
    repeat with thisSprite in gSpritesOnList
        puppetsprite (thisSprite, false)
    end repeat
    repeat with thisSprite in spritelist
        puppetsprite thisSprite, true
    end repeat
    set gSpritesOnList = spritelist
else
    repeat with thisSprite in spritelist
        puppetsprite thisSprite, true
    end repeat
    set gSpritesOnList = spritelist
end if
end

on spriteListOff
-- turns off all sprites on Current gSpritesonList
-- and re-initializes that global
global gSpritesOnList
if VoidP(gSpritesOnList) then set gSpritesOnList = []
repeat with thisSprite in gSpritesonList
    puppetsprite thisSprite, false
end repeat
set gSpritesOnList = []
end

on spriteson FirstSprite, LastSprite
-- turns on sprites in consecutive channels from
-- FirstSprite to LastSprite
global gSpritesOnList
if VoidP(gSpritesOnList) then set gSpritesOnList = []
if count(gSpritesOnList) > 0 then
    repeat with thisSprite in gSpritesOnList
        puppetsprite (thisSprite, false)
    end repeat
    set gSpritesOnList = []
    repeat with N = FirstSprite to LastSprite
        puppetsprite N, true
        add gSpritesonList, N
    end repeat
else
    set gSpritesOnList = []
    repeat with N = FirstSprite to LastSprite
        puppetsprite N, true
        add gSpritesonList, N
    end repeat
end if
end

```



```

on Spritesoff FirstSprite, LastSprite
  -- turns off sprites in consecutive channels from
  -- FirstSprite to LastSprite
  repeat with N = FirstSprite to LastSprite
    puppetsprite N, false
  end repeat
end

```

Movie Script87:WaitHandlers

```

on wait Howlong
  -- New Improved wait handler. doesn't reset timer
  -- every time it's called. be sure to StartTimer in
  -- "on StartMovie"
  set x = the timer
  put x into oldtime -- stores time
  repeat while (oldtime + Howlong) > x
    nothing
  set x = the timer
  end repeat
end wait

```

```

on waitPlus Howlong, doWhat
  -- Same as above handler except that allows passing
  -- of a handler to be executed during the wait
  -- Handler must be a string
  set x = the timer
  put x into oldtime -- stores time
  repeat while (oldtime + Howlong) > x
    do doWhat -- must be a string
    set x = the timer
  end repeat
end wait

```

```

on IgnoreMouseDowns
  if the mousedown then dontpassevent
end

```

```

on findDuplicateMemberNames whichCastLib
    put "looking for duplicates"
    set NumMems = the number of members of castLib whichCastLib
    set testedList = []
    sort testedList
    repeat with y = 1 to nummems
        -- put "looking"
        set DupeList = []
        if the type of member y of castLib whichCastLib = #empty then next repeat
        set whatName = the name of member y of CastLib whichCastLib
        if getOne(testedList,whatname) then next repeat
        add testedList whatname
        append dupelist,y
        repeat with x = 1 to nummems
            if the type of member x of castLib whichCastLib = #empty then next repeat
            if x = y then next repeat
            if the name of member x of castLib whichCastLib = whatName then
                append dupelist,x
            end if
        end repeat
        if count(dupelist) > 1 then
            put "Member"&&quote& whatName&quote&& "has these duplicates:"&&dupelist
        end if
    end repeat
    put "done looking"
end

```

Movie Script89

```
on PlaySoundsinCast whichCast, whereStart
-- plays every sound in the specified cast
-- and posts name and number in message window.
-- This allows quick check of sound and if the sound is looped
if not voidP(whereStart) then
    set y = wherestart
else
    set y = 1
end if
repeat with x = y to the number of members of castlib whichCast
    if the type of member x of castlib whichCast = #sound then
        put x && the name of member x of castlib whichCast
        puppetsound member x of castlib whichCast
        updatestage
        repeat while soundBusy(1)
            if mouseDown() = true then
                puppetsound 0
                exit
                return 0
            end if
        end repeat
    end if
end repeat
end
```

Score Script91

```
on exitFrame
    if soundBusy(1) then
        go to the frame
    end if
end
```

Score Script117

```
on mouseUp
  global gpausestatus, gLEDMan
  -- this button sends user to next frame for
  -- replay routine. This filters out user mouseDowns
  -- while the replay is occurring

  if gpausestatus <> "resume" then
    cursor 200
    set the mousedownscript to "dontpassevent"
    clearLEDDisplay gLEDMan
    go to frame The frame + 1
  end if
end
```

Score Script118

```
on exitFrame
  set rightlist = list("metal", "blues", "funk", "pop rock", "soft rock")
  set x = random(5)
  put getat(rightlist, x) into playmusic

  puppetsound 1.0
  preloadmember playmusic
  puppetsound 1. playmusic
  updatestage
end
```

Score Script120

```
on mouseUp
  global gpausestatus
  set gpausestatus = "resume"
  set the visible of sprite 46 to false
  go to marker (0)
  pause
end
```

Score Script121

```
on exitFrame
  -- uncomment this script for real play
  -- and deletete part underneath
  if soundbusy(2) then
    go to the frame
  else
    puppetsprite 2, false
    -- go to frame "rightmusic"
  end if
end
```

Score Script122

```
on exitFrame
  puppetsprite 2, false
  updatestage
end
```

Score Script133

```
on enterframe
  global gGame
  resetPlay gGame
  puppetsound 1, "listen"
  updatestage
  set sylNum = 0
  set oldTicks = the ticks
  repeat while soundbusy(1)
    if (sylNum < 2) and (the ticks > oldTicks + 5) then
      set the visible of sprite 9 to true
      updatestage
      wait 12
      set the visible of sprite 9 to false
      updatestage
      wait 12
      set sylnum = sylnum + 1
    end if
  end repeat
  repeatSounds gGame
  repeat while soundBusy(1)
    nothing
  end repeat
  go to frame the frame - 1
end

on exitframe
  global ghandcursor
  set the mouseDownscript to empty
  cursor ghandcursor
end
```

Score Script134

```
on mouseDown
  global gGame, gPauseStatus
  if gPauseStatus = "resume" then exit
  if the clickon = 4 then
    pass
  else
    userDrumHit gGame
  end if
end
```

Score Script135

```
on mouseDown
  global gGame, gPauseStatus
  if gPauseStatus = "resume" then exit
  if the clickon = 4 then
    pass
  else
    userDrumHit gGame
  end if
end
```

Score Script136

```
on mouseDown
  global gGame, gPauseStatus
  if gPauseStatus = "resume" then exit
  if the clickon = 4 then
    pass
  else
    userDrumHit gGame
  end if
end
```

Score Script137

```
on mouseUp
    StartUserDrumTime (300, "CheckUserHits gGame")
end
```

Score Script138

```
on mouseUp
    global gpausestatus, gLEDman
    set gpausestatus = "pause"
    set the visible of sprite 45 to false
    set the visible of sprite 46 to false
    set the visible of sprite 47 to false
    sound stop 1
    sound stop 2
    puppetsound 0
    continue
    clearLEDDisplay gLEDman
    StartUserDrumTime (300, "CheckUserHits gGame")
    startLEDDisplay (gLEDman, 300)
end
```

Script of Cast Member139

```
on mouseUp
    global gArmSprite
    puppetsprite gArmsprite, false
    go to frame "intro"
end
```

Script of Cast Member140

```
on mouseUp
    go to frame "black"
    go to movie "progress"
end
```

Score Script142

```
on exitFrame
  Global gNoter
  putUpNote gNoter
  puppetsound 2, "excellent"
end
```

Score Script144

```
on mouseUp
  repeat with x = 1 to 48
    puppetsprite x, false
  end repeat
  puppetsound (2, 0)
  puppetsound (1, 0)
  cursor 4
  go to frame "black"
  go to movie "progress"
end
```

Movie Script146:playAllSoundsAnd Segments

Score Script147

```
on exitFrame
  go to frame "test"
end
```

Score Script148

```
on exitFrame
  global gGame, gNoter
  init gNoter
  notespritesOn gNoter
  if the game of gGame = #drumSounds then
    go to frame "drum intro"
    exit
  else
    if the game of gGame = #IsoSounds then
      go to frame "drum intro"
      exit
    else
      if the game of gGame = #syllables then
        go to frame "drum intro"
        exit
      else
        if the game of gGame = #soundSegments then
          go to frame "sound heard"
          exit
        end if
      end if
    end if
  end if
end
```

Score Script149

```
on exitFrame
  go to frame "listen"
end
```

```

on PlaySoundNamesinField whichField
  set FieldLines = the number of lines in field whichField
  repeat with x = 1 to fieldLines
    set ItemCount = the number of items of line x of field whichField
    repeat with y = 1 to itemCount
      set sound = item y of line x of field whichField
      put sound
      puppetSound sound
      put sound && the number of member sound
      updatestage
      repeat while soundBusy(1)
        nothing
      end repeat
      wait 5
    end repeat
  end repeat
end

on findMissingsoundsinField whichField, toWhichField
  put "start looking for missing sounds"
  repeat with a = whichField to toWhichField
    if the type of member a = #field then
      put the name of member A into thisfield
      put thisfield

      set FieldLines = the number of lines in field thisfield
      repeat with x = 1 to fieldLines
        set ItemCount = the number of items of line x of field thisfield
        repeat with y = 1 to itemCount
          set sound = item y of line x of field thisfield
          if sound = "" then next repeat
          if the number of member sound = -1 then
            put sound&&"is Missing"
          end if
        end repeat
      end repeat
    end if
    put return
  end repeat

  put "finished looking for missing sounds"
end

on PlaySoundNamesandPieces whichField
  set FieldLines = the number of lines in field whichField
  repeat with x = 1 to fieldLines
    set ItemCount = the number of items of line x of field whichField
    repeat with y = 1 to itemCount
      set sound = item y of line x of field whichField
      put sound
      puppetSound sound
      updatestage
      repeat while soundBusy(1)
        nothing
      end repeat
      wait 5
      repeat with z = 1 to the number of words in sound
        put word z of sound
      end repeat
    end repeat
  end repeat
end

```

```

    puppetsound word z of sound
    updatestage
    repeat while soundBusy(1)
        nothing
    end repeat
    wait 5
end repeat
--      put sound && the number of member sound
updatestage

end repeat
end repeat
end

```

Movie Script156

```

on doMissingSoundsCheck
    set tempGame = new(script "IsoSound")
    set tempGame = 0
    set tempGame = new(script "syllables")
    FindMissingSoundsandSegments tempGame
    set tempGame = 0
    set tempGame = new(script "soundSegments")
    FindMissingSoundsandSegments tempGame
    set tempGame = 0
end

```

Movie Script161:testfoBadWords

```
on testForBadWords listOfWords
  set BadWordList = []
  set isBad = 0
  set numLines = the number of lines of field "badWords"
  repeat with x = 1 to numLines
    if numLines = "" then next repeat
    set tempList = []
    set numWords = the number of words in line x of field "badWords"
    repeat with y = 1 to numWords
      append tempList, word y of line x of field "badWords"
    end repeat
    append badwordList, tempList
  end repeat
  -- put badwordList
  set x = count(badwordList)
  repeat with y = 1 to x
    if getat(badwordList,y) = listOfWords then
      set isBad = 1
      return isBad
    end if
  end repeat
  return isbad
end
```

Movie Script170

```
on resetAllPurgePriorities
  -- step through all castlibs member by member and reset the
  -- purge priority of each member to 3
  put "Checking CastMember Purge Priorities"
  set castLibNums = the number of castLibs
  repeat with x = 1 to castLibNums
    set MemberNums = the number of members of castLib x
    repeat with y = 1 to memberNums
      if the purgePriority of member y of castLib x <> 3 then
        if the type of member y of castLib x = #sound then
          put "resetting PurgePriority of member "&y &&"of castLib"&&x
          set the purgePriority of member y of castLib x = 3
        end if
      end if
    end repeat
  end repeat
  put "Done Checking"
end
```

Movie Script171

```
on CheckPups
  set templist = []
  repeat with x = 1 to 48
    if the puppet of sprite x = true then
      append templist, x
    end if
  end repeat
  put templist
end
```

Score Script176

```
on exitFrame
  sound stop 1
  puppetsound 1.0

end
```

Score Script177

```
on exitFrame
  if soundbusy(2) then
    go to the frame
  end if
```

```
end
```

Score Script178

Score Script179

```
on exitFrame
  [
end
```

Score Script183

```
on exitFrame
```

```
end
```

Score Script184

```
on exitFrame
  if soundbusy(2) then
    go to the frame
  end if
```

```
end
```

Score Script185

Score Script186

```
on exitFrame
  global gGame
  -- check if prefs say we need to go to the
  -- next frame for prompt, otherwise go to intro and
  -- start again.
  if skipYesOrNo() then
    unloadSounds gGame
    resetAllPurgePriorities -- just in case
    go to frame "intro"
  end if
end
```

Script of Cast Member188

```
on mouseUp
  global gGame
  sound stop 1
  sound stop 2
  puppetsound 0
  repeat with x = 1 to 48
    puppetsprite x, false
    set the visible of sprite x to true
  end repeat
  if objectP(gGame) then unloadSounds gGame
  resetAllPurgePriorities -- just in case
  cursor 4
  go to frame "black"
  go to movie "progress"
end
```

Script of Cast Member189:pause.pict

```
--on mouseup
--  -- puppetSprite 45,true
--  -- set the member of sprite 45 to member "resume.pict"
--  -- updatestage
--  -- pause
--  set the visible of sprite 46 to false
--  pause
--end
```

Script of Cast Member190:resume.pict

```
on mouseUp
  global gpausestatus
  set gpausestatus = "pause"
  set the visible of sprite 45 to false
  set the visible of sprite 46 to false
  set the visible of sprite 47 to false
  sound stop 1
  sound stop 2
  puppetSound 0
  continue
end
```

```
on mouseUp
  global gpausestatus

  if the visible of sprite 47 = false then
    if gpausestatus = "pause" then
      set the visible of sprite 46 to true
    end if

    set the visible of sprite 45 to true
    set the visible of sprite 47 to true
  else
    set the visible of sprite 47 to false
    set the visible of sprite 45 to false
    set the visible of sprite 46 to false
  end if

  updatetage
  -- puppetsprite 45, true

end
```

Script of Cast Member197

```
on mouseUp
  global gpausestatus

  if the visible of sprite 47 = false then
    if gpausestatus = "pause" then
      set the visible of sprite 46 to true
    end if

    set the visible of sprite 45 to true
    set the visible of sprite 47 to true
  else
    set the visible of sprite 47 to false
    set the visible of sprite 45 to false
    set the visible of sprite 46 to false
  end if

  updatelstage
  -- puppetsprite 45, true

end
```

Script of Cast Member199

```
on mouseUp
  global gGame
  sound stop 1
  sound stop 2
  puppetsound 0
  repeat with x = 1 to 48
    puppetsprite x, false
    set the visible of sprite x to true
  end repeat
  if objectP(gGame) then unloadSounds gGame
  resetAllPurgePriorities -- just in case
  go to frame "black"
  go to movie "progress"
end
```

Score Script200

```
on exitFrame
  preload (the frame+1), (the frame+4)
  preload member 18
end
```

Parent Script1:PrintPlaceManager

```
Property pDate1Field, pDate1FieldSprite, pDate2Field, pDate2FieldSprite,
pCurEditableFieldSprite
Property pDateFormatField, pDateFormatFieldSprite, pDateFormat
Property pObDataViewPrefsMan, pDateFormatButtonSpriteList, pDateFormatChoice, pPrintMode

on new me
    set pDate1Field = "Date1Field"
    set pDate2Field = "Date2Field"
    set pDateFormatField = "DateFormatField"
    set pDateFormatFieldSprite = 18
    put " " into field pDate1Field
    put " " into field pDate2Field
    set pDate1FieldSprite = 14
    set pDate2FieldSprite = 16
    set pCurEditableFieldSprite = 0

    initDateFormatField me
    set pDateFormatButtonSpriteList = [7,8,9]
    return me
end

on drawPrintPlaceScreen me
    global gRadioButMan
    puppetSprite pDate1FieldSprite, true
    puppetSprite pDate2FieldSprite, true
    initRadioButtons gRadioButMan
end

on LeavePrintPlaceScreen me
    global gRadioButMan
    puppetSprite pDate1FieldSprite, false
    puppetSprite pDate2FieldSprite, false
    KillRadioButtons gRadioButMan
end

on shutDown me
    -- called when closing window to do cleanUp
    set the editable of sprite pDate1FieldSprite = false
    set the editable of sprite pDate2FieldSprite = false
    LeavePrintPlaceScreen me
end

on makeEditable me, whichSprite
    if pPrintMode <> 3 then exit
    if whichSprite = pCurEditableFieldSprite then exit
    set the editable of sprite pCurEditableFieldSprite = false
    set the editable of sprite whichSprite = true
    set pCurEditableFieldSprite = whichSprite
    if whichSprite = pDate1FieldSprite then
        set numChars = the number of chars of field pDate1Field
        set the selStart = numChars + 1
        set the selEnd = numChars + 1
    else
        set numChars = the number of chars of field pDate2Field
        set the selStart = numChars + 1
        set the selEnd = numChars + 1
    end
```

```

end if
end

on ChangeEditableField me
    if pPrintMode <> 3 then exit
    if pCurEditableFieldSprite = pDate1FieldSprite then
        set the editable of sprite pDate2FieldSprite = true
        set the editable of sprite pDate1FieldSprite = false
        set pCurEditableFieldSprite = pDate2FieldSprite
    else
        set the editable of sprite pDate1FieldSprite = true
        set the editable of sprite pDate2FieldSprite = false
        set pCurEditableFieldSprite = pDate1FieldSprite
    end if
    if pCurEditableFieldSprite = pDate1FieldSprite then
        set numChars = the number of chars of field pDate1Field
        set the selStart = numChars + 1
        set the selEnd = numChars + 1
    else
        set numChars = the number of chars of field pDate1Field
        set the selStart = numChars + 1
        set the selEnd = numChars + 1
    end if
end
end

```

```

on initDateFormatField me
    global gDataViewPrefsMan
    put getDateFormat(gDataViewPrefsMan) into pDateFormat
    case pDateFormat of
        0 : set text1 = "Month/Day/Year"
            set text2 = "12/25/98"
        1 : set text1 = "Day/Month/Year"
            set text2 = "25/12/98"
        2 : set text1 = "Year/Month/Day"
            set text2 = "98/12/25"
    end case
    put text1 into line 2 of field pDateFormatField
    put text2 into line 4 of field pDateFormatField
end

```

```

on drawNonUSDateScreen me
    --
    repeat with thisSprite in pDateFormatButtonSpriteList
        puppetSprite thisSprite, true
    end repeat
    set pDateFormatChoice = pDateFormat
    case pDateFormat of
        "0":set MDY = getAT(pDateFormatButtonSpriteList,1)
            set the member of sprite MDY = member "mdy down"
        "1":set DMY = getAT(pDateFormatButtonSpriteList,2)
            set the member of sprite DMY = member "DMY down"
        "2":set YMD = getAT(pDateFormatButtonSpriteList,3)
            set the member of sprite YMD = member "YMD down"
    end case
    updatestage
end

```

```

on LeaveNonUSDateScreen me

```



```

repeat with thisSprite in pDateFormatButtonSpriteList
  puppetSprite thisSprite, false
end repeat
end

on MakeDateFormatChoice me, whichSprite
  put word 1 of the name of the member of sprite whichSprite into pDateFormatChoice
  set ChoiceMemName = pDateFormatChoice && "down"
  set pDateFormatChoice = TransLateChoice(me, pDateFormatChoice)
  set the member of sprite whichSprite = member ChoiceMemName
  repeat with thisSprite in pDateFormatButtonSpriteList
    if thisSprite = whichSprite then next repeat
    set theName = word 1 of the name of the member of sprite thisSprite
    set the member of sprite thisSprite = member theName
  end repeat
  updatestage
end

on CancelDateFormatChoice me
  put 0 into pDateFormatChoice
  repeat with thisSprite in pDateFormatButtonSpriteList
    set theName = word 1 of the name of the member of sprite thisSprite
    set the member of sprite thisSprite = member theName
  end repeat
  updatestage
end

on confirmDateFormatChoice me
  global gDataViewPrefsMan
  put value(pDateFormatChoice) into dateFormat
  setDateFormat (gDataViewPrefsMan, dateFormat)
  set pDateFormat = dateFormat
  initDateFormatField me
end

on TransLateChoice me, aString
  case astring of
    "mdy": return 0
    "dmy": return 1
    "ymd": return 2
  end case
end

on updatePrintMode me, PrintMode
  set pPrintMode = PrintMode
  set PrintText = "PrintMode"&printMode&&"text"
  put field PrintText into field "helpfield"
  if PrintMode = 3 then
    set the border of member pDate1Field = 1
    set the border of member pDate2Field = 1
    set the editable of sprite pDate1FieldSprite = true
    set pCurEditableFieldSprite = pDate1FieldSprite
    set numChars = the number of chars of field pDate1Field
    set the selStart = numChars + 1
    set the selEnd = numChars + 1
  else
    put " " into field pDate1Field
    put " " into field pDate2Field
    set the editable of sprite pDate1FieldSprite = false
  end
end

```



```

    set the editable of sprite pDate2FieldSprite = false
    set the border of member pDate1Field = 0
    set the border of member pDate2Field = 0
    set pCurEditableFieldSprite = 0
end if

-- other stuff tBA
end

on startRangePrinting me
    -- first start at printing range

    global gPrintMan
    put field pDate1Field into date1
    put field pDate2Field into date2
    if the number of words of date2 > 0 then -- something 's there
        set doAlert = CheckDateFormat (date1) + CheckDateFormat (date2)
    else
        set doAlert = CheckDateFormat (date1)
    end if

    if doAlert then
        putUpAlert #DateFormatError
        exit
    end if

    if the number of words of date2 > 0 then
        PrintDataRange gPrintMan , date1,date2
    else
        PrintDataRange gPrintMan , date1
    end if
end

on doPrint me
    case pPrintMode of
        1:printOneSession me
        2:printAllData me
        3: startRangePrinting me
    end case
end

on PrintOneSession me
    global gPrintMan
    PrintOneSession(gPrintMan)
end

on PrintAllData me
    global gPrintMan
    printAllData (gPrintMan)
end

on doNoDataInRangeError me
    global gPrintMan
    put field pDate1Field into date1
    put field pDate2Field into date2
    if the number of words of date2 < 1 then
        put the date into date2
    end if
end

```

```

end if
put sendPrintSpecs (gPrintMan) into aList
set name = getAT(aList,1)
set game = getAT(aList,2)
put game into word 4 of line 3 of field "NoDataInRangeAlerttext"
put name into word 3 of line 4 of field "NoDataInRangeAlerttext"
put date1 into word 1 of line 6 of field "NoDataInRangeAlerttext"
put date2 into word 3 of line 6 of field "NoDataInRangeAlerttext"
-- no data with the dates range so tell the user so
putUpAlert #NoDataInRange
end

on doNonStandardDateFormatError me
    putUpAlert #NonStandardDateFormat
end

```

Parent Script2:RadioButtonManager

Property pButtonSpriteList, pCurrentButton, pDefaultButtonSprite,
pButtonName, pButtonOnName

```
on new me, spriteList
    set pButtonSpriteList = spriteList -- list of 3 sprite channels
    set pButtonName = "printModeBut"
    set pButtonOnName = "printModeBut On"
    set pDefaultButtonSprite = getAT(pButtonSpriteList,1)
    return me
end

on initRadioButtons me
    global gPrintPlaceMan
    repeat with thisSprite in pButtonSpriteList
        puppetSprite thisSprite , true
    end repeat
    if pCurrentButton = getAT(pButtonSpriteList,3) then -- we're back here from an alert
        set the member of sprite pCurrentButton = member pButtonOnName
        updatePrintMode gPrintPlaceMan, 3
    else -- we're here for the first time
        set the member of sprite pDefaultButtonSprite = member pButtonOnName
        set pCurrentButton = pDefaultButtonSprite
        updatePrintMode gPrintPlaceMan, 1
    end if
end

on KillRadioButtons me
    repeat with thisSprite in pButtonSpriteList
        puppetSprite thisSprite , false
    end repeat
end

on doRadioAction me, whichSprite
    global gPrintPlaceMan
    if whichSprite = pCurrentButton then exit
    set pCurrentButton = whichSprite
    repeat with x = 1 to 3
        set thisSprite = getAT(pButtonSpriteList, x)
        if thisSprite = whichSprite then
            set the member of sprite thisSprite = member pButtonOnName
        else
            set the member of sprite thisSprite = member pButtonName
        end if
    end repeat
    set PrintMode = getPos(pButtonSpriteList, whichSprite)
    updatePrintMode(gPrintPlaceMan, PrintMode)
end
```

Parent Script3:DataViewPrefsManager

Property ancestor, pDateFormat, pCastLibName

on new me

global gRecordKeeper

set ancestor = gRecordKeeper

set pCastLibName = the castLibName of ancestor

return me

end

on getDateFormat me

-- returns 0,1,or 2 which is stored in line 1 of

-- dataView Prefs cast. If cast not there it makes on

-- and returns 0 to start as a default

-- 0 means MM/DD/YY

-- 1 means DD/MM/YY

-- 2 means YY/MM/DD

openrecords me

if the number of member "dataViewPrefs" of castLib pCastLibName = -1 then

-- no prefs so make new one

new #field, member 5 of castLib pCastLibName

put 0 into line 1 of field 5 of castLib pCastLibName

set the name of member 5 of castLib pCastLibName = "DataViewPrefs"

set pDateFormat = 0

saveRecords me

CloseRecords me

return 0

else

set pDateFormat = line 1 of field "DataViewPrefs" of castLib pCastLibName

closeRecords me

return pDateFormat

end if

end

on setDateFormat me, formatToSave

-- stores 0,1,or 2 which is stored in line 1 of

-- dataView Prefs cast. If cast not there it makes on

-- and returns 0 to start as a default

-- 0 means MM/DD/YY

-- 1 means DD/MM/YY

-- 2 means YY/MM/DD

openrecords me

if the number of member "dataViewPrefs" of castLib pCastLibName = -1 then

-- no prefs so make new one

new #field, member 5 of castLib pCastLibName

set the name of member 5 of castLib pCastLibName = "DataViewPrefs"

end if

put formatToSave into line 1 of field 5 of castLib pCastLibName

set pDateFormat = formatToSave

saveRecords me

CloseRecords me

end

Score Script6

Script of Cast Member14:PrintRange

```
on mouseUp
    global gPrintPlaceMan
    startRangePrinting gPrintPlaceMan
end
```

Score Script15

```
on mouseDown
    global gPrintMan, gPrintPlaceMan, gVoid, gRadioButMan
    if legalButtonHandler() then
        shutDown gPrintPlaceMan
        set gPrintPlaceMan = gVoid
        set gRadioButMan = gVoid
        closePrintPlace gPrintMan
    end if
end
```

Score Script19

```
on exitFrame
    go to the frame
end

on keyUp
    global gPrintPlaceMan
    if the keyCode = 48 then -- TAB
        ChangeEditableField gPrintPlaceMan
    else if the keycode = 36 then --- space then
        doPrint gPrintPlaceMan
    end if
end
```

Movie Script20

```
on startMovie
  global gPrintPlaceMan, gRadioButMan

  if not objectP(gPrintPlaceMan) then
    set gPrintPlaceMan = new(script "printPlaceManager")
  end if

  if not objectP(gRadioButMan) then
    set gRadioButMan = new(script "RadioButtonManager" [20,21,22])
    -- list is radio button sprites, first one is default
  end if

  initFields

  go to "printPlace"
end

on stopMovie
  global gPrintPlaceMan, gVoid, gRadioButMan
  set gPrintPlaceMan = gVoid
  set gRadioButMan = gVoid
end

on initFields
  global gPrintMan
  put the date into field "todaysDate"
  set the textsize of field "todaysDate" = 36
  put field "PrintThisSession refill" into field "PrintThisSession"
  put field "PrintAllData refill" into field "PrintAllData"
  put field "RangeTitle refill" into field "RangeTitle"
  set PrintSpecList = sendPrintSpecs(gPrintMan) -- returns strings of user, game date
  set user = getAT(printSpecList,1)
  set game = getAT(printSpecList,2)
  set theDate = getAT(printSpecList,3)

  put theDate&"." into word 6 of field "PrintThisSession"
  put game into word 3 of field "PrintThisSession"
  put user&"'s" into word 2 of field "PrintThisSession"

  put game into word 9 of line 1 of field "PrintAllData"
  put user&"'s" into word 2 of line 1 of field "PrintAllData"

  put game into word 3 of field "rangeTitle"
  put user&"'s" into word 2 of field "rangeTitle"

  put field "Printmodel text" into field "helpfield"
  -- set the border of member "helpfield" = 1
  set the margin of member "helpfield" = 4
end
```

Score Script21:DateFieldSpriteScripts

```
on mouseUp
    global gPrintPlaceMan
    set mySprite = the clickOn
    makeEditable gPrintPlaceMan, mySprite
end

on KeyDown
    global gPrintPlaceMan
    case the keycode of
        36: ChangeEditablefield gPrintPlaceMan -- return
        51: pass -- delete
        otherwise checkForLegalKeys
    end case
end

on checkForLegalKeys
    if "1234567890/" contains the key then pass
end
```

Score Script22

```
on enterFrame
    global gPrintPlaceman
    drawPrintPlaceScreen gPrintPlaceman
    initNoDataInRangeAlert
end
```

Score Script23

```
on exitFrame
    go to the frame
end
```

Score Script33

```
on mouseUp
  global gPrintPlaceMan
  set mySprite = the ClickOn
  MakeDateFormatChoice gPrintPlaceMan, mySprite
end
```

Score Script34

Score Script35

```
on enterFrame
  global gPrintPlaceMan
  drawNonUSDateScreen gPrintPlaceMan
end
```

Score Script36

```
on mouseUp
  global gPrintPlaceMan
  LeavePrintPlaceScreen gPrintPlaceMan
  go to "NonUsDates"
end
```

Score Script37

```
on mouseUp
  global gPrintPlaceMan
end
```

Score Script38

```
on exitFrame  
  go to the frame  
end
```

Script of Cast Member42:ConfirmFormatChoice

```
on mouseUp  
  global gPrintPlaceMan  
  confirmDateFormatChoice gPrintPlaceMan  
  LeaveNonUSDateScreen gPrintPlaceMan  
  go to "printPlace"  
end
```

Script of Cast Member43:ExitToPrintPlace

```
on mouseUp  
  global gPrintPlaceMan  
  LeaveNonUSDateScreen gPrintPlaceMan  
  go to "printPlace"  
end
```

Movie Script44

```
on CheckDateFormat dateToCheck
  put the itemdelimiter into OldItemDlmtr
  set the itemdelimiter to "/"
  if the number of items in dateToCheck <> 3 then -- wrong
    set the itemdelimiter = OldItemDlmtr
    return 1
  else
    repeat with x = 1 to 3
      if item x of dateToCheck = "" then
        set the itemdelimiter = OldItemDlmtr
        return 1
      end if
    end repeat
    set the itemdelimiter = OldItemDlmtr
    return 0
  end if
end
```

Movie Script45

```
on putUpAlert whichKind
  global gPrintPlaceMan
  if whichKind = #DateFormatError then
    LeavePrintPlaceScreen gPrintPlaceMan
    go to "DateFormatAlert"
    exit
  end if
  if whichKind = #NoDataInRange then
    LeavePrintPlaceScreen gPrintPlaceMan
    go to "NoDataInRangeAlert"
    exit
  end if
  if whichKind = #NonStandardDateFormat then
    LeavePrintPlaceScreen gPrintPlaceMan
    go to "NonStandardDateAlert"
    exit
  end if
end
```

Score Script46

```
on mouseUp
  go to "printPlace"
end
```

Score Script47

```
on mouseDown
  global gRadioButMan
  set mySprite = 20
  doRadioAction gRadioButMan, mySprite
end
```

Movie Script48

```
on initNoDataInRangeAlert
  put field "NoDataInRangeAlerttext refill" into field "NoDataInRangeAlerttext"
  -- set the fontsize of line 1 of field "NoDataInRangeAlerttext" = 36
  -- set x = the number of lines in field "NoDataInRangeAlerttext"
  -- repeat with y = 3 to x
  --   set the fontsize of line y of field "NoDataInRangeAlerttext" = 24
  -- end repeat
end
```

Score Script49

```
on mouseDown
  global gRadioButMan
  set mySprite = the ClickOn
  doRadioAction gRadioButMan, mySprite
end
```

Score Script50

```
on mouseUp
    global gPrintPlaceMan
    PrintOneSession gPrintPlaceMan
end
```

Score Script52

```
on mouseUp
    global gPrintPlaceMan
    PrintAllData gPrintPlaceMan
end
```

Score Script53

Score Script54

```
on exitFrame
    go to the frame
end

on mouseUp
    go to "printPlace"
end
```

Score Script59

```
on mouseDown
    if legalbuttonHandler() then
        global gPrintPlaceMan
        doPrint gPrintPlaceMan
    end if
end
```

Score Script60

```
on mouseDown
  global gRadioButMan
  set mySprite = 21
  doRadioAction gRadioButMan, mySprite
end
```

Score Script64

```
on mouseDown
  global gRadioButMan
  set mySprite = 22
  doRadioAction gRadioButMan, mySprite
end
```

Movie Script67:LegalButtonHandler

on LegalButtonHandler

```
set lRollover = TRUE
set clickSprite = the clickon
set lButtonUpName = the name of member (the member of sprite clickSprite)
set lButtonDownName = lButtonUpName & "Down"
set the member of sprite clickSprite = member lButtonDownName
updatestage

repeat while the stillDown = TRUE
  if rollover (clickSprite) = TRUE then
    set lRollover = TRUE
    set the membernum of sprite clickSprite = the number of member lButtonDownName
    updatestage
  else
    set lRollover = FALSE
    set the membernum of sprite clickSprite = the number of member lButtonupName
    updatestage
  end if
end repeat

if lRollover = FALSE then return 0

set the membernum of sprite clickSprite = the number of member lButtonupName
updatestage
return 1
```

end LegalButtonHandler

Score Script68

```
on mouseUp
  go to "printPlace"
end
```

Script of Cast Member135:exitPrintingxxx

```
on mouseUp
    global gPrintMan, gPrintPlaceMan, gVoid, gRadioButMan
    shutDown gPrintPlaceMan
    set gPrintPlaceMan = gVoid
    set gRadioButMan = gVoid
    closePrintPlace gPrintMan
end
```


Movie Script4

```
on startMovie
  global
  gcharlist, gthename, gchartlist, goldscoringlevel, thegame, grecordkeeper, gorderList, gbigchartlist, chartpro

  put getprostatus(grecordkeeper) into chartpro
  -- set chartpro = "0" --Test

  put getUserSkipGameList(grecordkeeper, gthename) into skiplist
  set skipbuttonlist = [46, 48, 47, 45, 43, 44]
  repeat with x = 1 to 6
    put getat(skiplist, x) into skipgame
    put getat(skipbuttonlist, x) into skipwhatsprite
    if skipgame = 1 then
      set the visible of sprite skipwhatsprite to false
    end if
  end repeat

  cursor 4
  register(xtra "printomatic", "POMX153-501-02580")
  puppetsound 1, 0
  puppetsound 2, 0

  set gorderList = list()
  set gorderList =
  ["chartKaty", "chartEggs", "chartFrog", "chartRappers", "ChartBalloons", "ChartCoal"]
  set gchartlist = list()

  set the font of member "UserName" = "helvetica"
  set the fontsize of member "UserName" = 14
  set the forecolor of member "UserName" = the forecolor of member "color"

  put gthename into field "username"

  put loadchart(grecordkeeper, gthename) into gchartlist
  put getGameLevelLists(grecordkeeper, gthename) into gbigchartlist
  put "gbigchartlist =" & gbigchartlist
  -- set gbigchartlist = [[3, 2, 1, 1, 3, 1], [1, 0, 0, 1, 1, 0, 0], [1, 0], [1, 0, 1, 0], [2, 1, 0, 0], [1, 0, 1, 0, 1, 1, 1]] ---TEST

  set x = value(thegame)

  if thegame <> 0 then
    put getat(gorderList, x) into curr
    deleteat(gorderList, x)
    append(gorderList, curr)
  end if

  if goldscoringlevel = 0 then
    set goldscoringlevel = 1
```

```

end if

-- set goldscoringlevel = 5 --TEST
repeat with x = 37 to 42
    set the forecolor of sprite x = the forecolor of member "red"
end repeat
set the forecolor of sprite 38 = the forecolor of member "purple"

pickplayball

end

on stopmovie
    global gdatasavepath
    repeat with x = 3 to 8
        puppetsprite x, false
    end repeat
    if the runmode = "author" then
        put "x" into field "UserName"
    end if

    set the font of member "UserName" = "helvetica"
    set the fontsize of member "UserName" = 14
    set the forecolor of member "UserName" = the forecolor of member "color"
    if fileexists(gdatasavepath & "temp.bmp") = 0 then
        deletefile(gdatasavepath & "temp.bmp")
    end if
end

end

on buttonDownhandler
    put the clickon into x
    set buttonName = word 1 of the name of member (the member of sprite x)

    set downbutton = buttonName && "down"
    set upbutton = buttonName && "up"
    set the member of sprite x to member downbutton
    updatestage
    repeat while the mousedown
        if rollover (x) then
            set the member of sprite x to member downbutton
            updatestage
        else
            set the member of sprite x to member upbutton
            updatestage
        end if
    end repeat
end

end

on buttonUPhandler
    global gtheName, theGame
    put the clickon into x

```

```

put word 1 of the name of member the mousecast into buttonName
set downbutton = buttonName && "down"
set upbutton = buttonName && "up"
if the name of member the mousecast = downbutton then
    set the member of sprite x to member upbutton
    updatestage

```

```

-- sound stop 1
-- puppetsound 0
cursor 4

```

```

repeat with x = 1 to 48
    puppetsprite x, false
end repeat

```

```

if field "UserName" <> "" then
    put field "UserName" into theName
    set gtheName = theName

```

```

case (buttonname) of
    "clown" : put "5" into theGame
    go to frame "mac"
    go to movie "Karloona"
    exit

```

```

    "coal" : put "6" into theGame
    go to frame "black"
    go to movie "coal8"
    "rapper" : put "4" into theGame
    go to frame "black"
    go to movie "rappers8"

```

```

    "katy" : put "1" into theGame
    go to frame "black"
    go to movie "Katy8"

```

```

    "frog" : put "3" into theGame
    go to frame "black"
    go to movie "Rhyme8"

```

```

    "farmer" : put "2" into theGame
    go to frame "black"
    go to movie "Eggs8"
end case

```

```

repeat with g = 3 to 8
    puppetsprite g, false
end repeat

```

```

else
    alert "You must first select a player"
    go to marker (0)

```

```

end if
end if

```

```

end

```

```

on loadchart --not currently used
    global gtheName, gchartlist
    set gchartlist = list()

```



```

-- set goldscoringlist = [0,0,0,0,0,0] --Test
put getat(gbigchartlist,1) into gscoringlist

set katylist = list()
put getat(gchartlist,1) into katylist
put getat(katylist,1) into katyscoringlevel
put getat(katylist,2) into katyhighlevel

set levelkatylist = [[0,4],[5,8],[9,12],[13,16],[17,20],[21,24],
[25,28],[29,32],[33,36],[37,40],[41,44],[45,48],[49,52],[53,56],
[57,60]]

put count(levelkatylist) into listNum
repeat with y = 1 to listNum
  put getat(levelkatylist,y) into checklevel
  put getat(checklevel,1) into d
  put getat(checklevel,2) into e
  if (katyscoringlevel >= d) and (katyscoringlevel <= e) then
    set scoringlevel = y
  end if
  if (katyhighlevel >= d) and (katyhighlevel <= e) then
    set katyhighlevel = y
  end if
  if thegame = "1" then
    if (goldscoringlevel >= d) and (goldscoringlevel <= e) then
      set goldscoringlevel = y
    end if
  end if
end repeat

put "katyscoringlevel = " & scoringlevel
put "katyhighlevel = " & katyhighlevel

--
if katyscoringlevel > 15 then
  set katyscoringlevel = 15
end if
if katyhighlevel > 15 then
  set katyhighlevel = 15
end if

-- set scoringlevel = 11 -- TEST
-- set goldscoringlevel = 1 -- TEST

GiveMeBalls

--
if chartpro = "0" then
  set katyVspritelist =
list(1000,309,309,275,275,241,241,207,207,173,173,139,139,105,105)
  set katyHspritelist =
list(1000,344,374,344,374,344,374,344,374,344,374,344,374,344,374)

  put getat(katyVspritelist,katyhighlevel) into y
  put getat(katyHspritelist,katyhighlevel) into x
  set the locV of sprite 40 to y
  set the locH of sprite 40 to x
  set the visible of sprite 40 to true

```



```

-- updatestage
else
  set the loc of sprite 40 to point(1000,1000)
  set the visible of sprite 40 to false
end if

end

on chartEggs
  global
  gchartlist,thegame,goldscoringlevel,gsfx,scoringlevel,highlevel,minlist,maxlist,spritelist,monsterlist,~
  ballchart,ballmaster,goldscoringlist,gscoringlist,newscorelist,whatgame,gbigchartlist,chartpro,specialsound
  set specialsound = "farmerWin"
  set ballchart = [{"e_1_0", "e_1_1"}, {"e_2_0", "e_2_1", "e_2_2", "e_2_3", "e_2_4"}]
  set ballmaster = [1,1,1,1,2,2,2]
  set maxlist = [1,1,1,1,4,4,4]
  set minlist = [0,0,0,0,0,0,0]
  set spritelist = [10,11,12,13,14,15,16]
  set whatgame = "2"
  put getat(gbigchartlist,2) into gscoringlist
  set egglist = list()
  put getat(gchartlist,2) into egglist
  put getat(egglist,1) into Eggscoringlevel
  put getat(egglist,2) into Egghighlevel
  if Egghighlevel = 0 then
    set Egghighlevel = 1
  end if

  set levelegglist = [[0,3],[4,11],[12,20],[21,30],[31,37],[38,44],[45,51],[52,58],[59,65],[66,72],[73,79],[80,86],[87,93],[94,100],[101,107],[108,114],[115,116]]

  put count(levelegglist) into listNum
  repeat with y = 1 to listNum
    put getat(levelegglist,y) into checklevel
    put getat(checklevel,1) into d
    put getat(checklevel,2) into e
    if (Eggscoringlevel >= d) and (Eggscoringlevel <= e) then
      set scoringlevel = y
    end if
    if (Egghighlevel >= d) and (Egghighlevel <= e) then
      set Egghighlevel = y
    end if
    if thegame = "2" then
      if (goldscoringlevel >= d) and (goldscoringlevel <= e) then
        set goldscoringlevel = y
      end if
    end if
  end repeat

  put "Eggscoringlevel = "& scoringlevel

```



```

put "Egghighlevel" = "& Egghighlevel

if Eggscoringlevel > 17 then
  set Eggscoringlevel = 17
end if
if Egghighlevel > 17 then
  set Egghighlevel = 17
end if

--
--
GiveMeBalls
--
if chartpro = "0" then
  set EggVspritelist =
list(1000,309,309,275,275,241,241,207,207,173,173,139,139,105,105,71,71)
  set EggHspritelist =
list(1000,547,577,547,577,547,577,547,577,547,577,547,577,547,577,547,577)
  put getat(EggVspritelist,Egghighlevel) into y
  put getat(EggHspritelist,Egghighlevel) into x
  set the locV of sprite 42 to y
  set the locH of sprite 42 to x
  set the visible of sprite 42 to true
  -- updatestage
else
  set the loc of sprite 42 to point(1000,1000)
  set the visible of sprite 42 to false
end if

end

on test
  set levelegglist = {[0,1,2,3],[4,5,6,7,8,9,10,
11],[12,13,14,15,16,17,18,19,20]}
  set Eggscoringlevel = 6
  repeat with x in levelegglist
    if Eggscoringlevel = x then
      put getat(levelegglist,x) into y
      put "y = " & y
    end if
  end repeat
end

on getcastnames
  global wholelist
  set templist = []
  set wholelist = []
  repeat with x = 301 to 302
    append templist, the name of member x
  end repeat
  append wholelist,templist
  set templist = []
  repeat with x = 311 to 315
    append templist, the name of member x
  end repeat
  append wholelist,templist
  set templist = []

  put wholelist

```

end

on chartFrog

global

gchartlist, thegame, goldscoringlevel, gsfx, scoringlevel, highlevel, minlist, maxlist, spritelis
t, monsterlist, ~

ballchart, ballmaster, goldscoringlist, gscoringlist, newscorelist, whatgame, gbigchartlist, cha
rtpro, specialsound

set specialsound = "FrogWin"

set ballchart = [{"f_1_0", "f_1_1", "f_1_2", "f_1_3", "f_1_4", "f_1_5"}, {"f_2_0",
"f_2_1", "f_2_2", "f_2_3", "f_2_4", "f_2_5", "f_2_6"}]

set ballmaster = [1,2]

set whatgame = "3"

set maxlist = [5,6]

set minlist = [0,0]

set spritelist = [17,18]

-- set goldscoringlist = [0,0] --Test

put getat(gbigchartlist,3) into gscoringlist

set froglist = list()

put getat(gchartlist,3) into froglist

put getat(froglist,1) into scoringlevel

put getat(froglist,2) into froghighlevel

if froghighlevel = 0 then

set froghighlevel = 1

end if

put "Frogscoringlevel = "& scoringlevel

put "Froghighlevel = "& froghighlevel

if scoringlevel > 12 then

set scoringlevel = 12

end if

if froghighlevel > 12 then

set froghighlevel = 12

end if

-- set scoringlevel = 11 -- TEST

-- set goldscoringlevel = 1 -- TEST

--

GiveMeBalls

--

if chartpro = "0" then

set FrogVspritelist = list(1000,309,309,275,275,241,1,207,207,173,173,139)

set FrogHspritelist = list(1000,447,477,447,477,447,7,447,477,447,477,447)

put getat(FrogVspritelist,froghighlevel) into y

put getat(FrogHspritelist,froghighlevel) into x

set the locV of sprite 41 to y

set the locH of sprite 41 to x

set the visible of sprite 41 to true

-- updatestage

else

set the loc of sprite 41 to point(1000,1000)

set the visible of sprite 41 to false

end if

end

```

on chartRappers
  global
  gchartlist,thegame,goldscoringlevel,gsfx,scoringlevel,highlevel,minlist,maxlist,spritel
  t,monsterlist,-
  ballchart,ballmaster,goldscoringlist,gscoringlist,newscorelist,whatgame,gbigchartlist,c
  rtpro,specialsound
  set specialsound = "rapWin"
  set ballchart = [{"r_1_0", "r_1_1", "r_1_2", "r_1_3", "r_1_4", "r_1_5", "r_1_6"},
  [{"r_2_0", "r_2_1", "r_2_2"}]
  set ballmaster = {1,1,2,2}
  set maxlist = {6,6,2,2}
  set minlist = {0,0,0,0}
  set whatgame = "4"
  set spritelist = {19,20,21,22}
  put getat(gbigchartlist,4) into gscoringlist
  set raplist = list()
  put getat(gchartlist,4) into raplist
  put getat(raplist,1) into scoringlevel
  put getat(raplist,2) into Raphighlevel
  if Raphighlevel = 0 then
    set Raphighlevel = 1
  end if

  put "Rapscoringlevel" = "& scoringlevel
  put "Raphighlevel" = "& Raphighlevel

  if scoringlevel > 17 then
    set scoringlevel = 17
  end if
  if Raphighlevel > 17 then
    set Raphighlevel = 17
  end if

  --
  givemeballs
  --
  if chartpro = "0" then
    set RapVspritelist =
    list(1000,309,309,275,275,241,241,207,207,173,173,139,139,105,105,71,71)
    set RapHspritelist =
    list(1000,243,273,243,273,243,273,243,273,243,273,243,273,243,273,243,273)
    put getat(RapVspritelist,Raphighlevel) into y
    put getat(RapHspritelist,Raphighlevel) into x
    set the locV of sprite 39 to y
    set the locH of sprite 39 to x
    set the visible of sprite 39 to true
    -- updatestage
  else
    set the loc of sprite 39 to point(1000,1000)
    set the visible of sprite 39 to false
  end if

end

on chartBalloons
  global
  gchartlist,thegame,goldscoringlevel,gsfx,scoringlevel,highlevel,minlist,maxlist,spriteli
  t,monsterlist,-

```

```
ballchart,ballmaster,goldscoringlist,gscoringlist,newscorelist,whatgame,gbigchartlist,chartpro,specialsound
```

```
set specialsound = "karloonwin"
set ballmaster = [1,1,1,1]
set ballchart = [{"Karloon0","Karloon1","Karloon2"}]
set newscorelist = [0,0,0,0]
-- set goldscoringlist = [0,2,0,1] --TEST VARIABLE
put getat(gbigchartlist,5) into gscoringlist
set whatgame = "5"
set balloonlist = list()
put getat(gchartlist,5) into balloonlist
put getat(balloonlist,1) into balloonscorelevel
put getat(balloonlist,2) into balloonhighlevel
if balloonhighlevel = 0 then
  set balloonhighlevel = 1
end if
set maxlist = [2,2,2,2]
set minlist = [0,0,0,0]
set spritelist = [23,24,25,26]
set monsterlist = [1,1,2,2,3,3,4,4]
set levelballoonlist =
[[0,3],[4,6],[7,12],[13,18],[19,24],[25,30],[31,34],[35,38],[39,40]]
put count(levelballoonlist) into listNum
repeat with y = 1 to listNum
  put getat(levelballoonlist,y) into checklevel
  put getat(checklevel,1) into d
  put getat(checklevel,2) into e

  if (balloonscorelevel >= d) and (balloonscorelevel <= e) then
    set scorelevel = y
  end if

  if (balloonhighlevel >= d) and (balloonhighlevel <= e) then
    set balloonhighlevel = y
  end if

  if thegame = "5" then
    if (goldscorelevel >= d) and (goldscorelevel <= e) then
      set goldscorelevel = y
    end if
  end if
end repeat
-- set scorelevel = 8 -- TEST
-- set goldscorelevel = 1 -- TEST
-----
put " balloonscorelevel =" & scorelevel
put " goldscorelevel =" & goldscorelevel
```

GivemeBalls

```
if chartpro = "0" then
  set balloonVspritelist = list(1000,309,275,241,207,173,139,105,71)
  set balloonHspritelist = list(1000,63,63,63,63,63,63,63,63)
  put getat(balloonVspritelist,balloonhighlevel) into y
  put getat(balloonHspritelist,balloonhighlevel) into x
  set the locV of sprite 37 to y
  set the locH of sprite 37 to x
  set the visible of sprite 37 to true
  -- updatestage
else
  set the loc of sprite 37 to point(1000,1000)
  set the visible of sprite 37 to false
```

```

end if
end

on GivemeBalls
  global
  gchartlist, thegame, goldscoringlevel, gsfx, scoringlevel, highlevel, minlist, maxlist, spritel
  t, monsterlist,
  ballchart, ballmaster, goldscoringlist, gscoringlist, newscorelist, whatgame, specialsound, gt
  chartlist, specialsound
  case(whatgame) of
    "1":set the visible of sprite 40 to false
    "2":set the visible of sprite 42 to false
    "3":set the visible of sprite 41 to false
    "4":set the visible of sprite 39 to false
    "5":set the visible of sprite 37 to false
    "6":set the visible of sprite 38 to false
  end case

  if thegame = whatgame then
    put count(maxlist) into ListNum

    repeat with x = 1 to ListNum
      put getat(ballchart, getat(ballmaster, x)) into whatchart
      set the member of sprite getat(spritelist, x) to member
      getat(whatchart, (getat(goldscoringlist, x)+1))
    end repeat
    updatestage
    starttimer
    repeat while the timer <120
      end repeat
    ---
    if goldscoringlevel < scoringlevel then

      put count(maxlist) into ListNum
      repeat with x = 1 to ListNum
        set targetsprite = getat(spritelist, x)
        set Num1 = getat(goldscoringlist, x)+1
        set Num2 = getat(gscoringlist, x)+1
        put getat(ballchart, getat(ballmaster, x)) into whatchart

        if Num1 < Num2 then
          repeat with y = Num1 to Num2 --- repeat with y = 1 to 3
            if y > Num1 then
              set the member of sprite targetsprite to member getat(whatchart, y)
              put getat(whatchart, y)
              updatestage
              puppetsound 1, gsfx
              updatestage
              repeat while soundbusy(1)
                end repeat
            end if
          end repeat
        end if
      end repeat
    end if
  end if
end

```



```

end repeat
else if goldscoringlevel > scoringlevel then

repeat with x = ListNum down to 1
  put x
  set targetsprite = getat(spritelist,x)
  set Num1 = getat(goldscoringlist,x)+1 --2
  set Num2 =getat(gscoringlist,x)+1 ----1
  put getat(ballchart,getat(ballmaster,x)) into whatchart
  if Num1 > Num2 then
    repeat with y = Num1 down to Num2 --- repeat with y = 2 to 1
      if y < Num1 then
        set the member of sprite targetsprite to member getat(whatchart,y)
        updatestage
        puppetsound 1,"wrong.aif"
        updatestage
        repeat while soundbusy(1)
        end repeat
      end if
    end repeat
  end if
end repeat
--checks to see if game is completed and plays sound
if gscoringlist = maxlist then
  puppetsound 1,0
  puppetsound 1, specialsound
  updatestage
  repeat while soundbusy(1)
  end repeat

end if
--checks if all games are completed and plays sound
if gbigchartlist = [[3,3,1,1,3,3], [1,1,1,1,4,4,4], [5,6], [6,6,2,2], [2,2,2,2],
[1,1,1,1,1,1,4]] then
  puppetsound 1,0
  puppetsound 1, "final"
  updatestage
  repeat while soundbusy(1)
  end repeat

end if

--plays backround music when done
puppetsound 1,0
puppetsound 1, "main track"
updatestage

else
  put count(maxlist) into ListNum

  repeat with x = 1 to ListNum
    put getat(ballchart,getat(ballmaster,x)) into whatchart
    set the member of sprite getat(spritelist,x) to member
    getat(whatchart,(getat(gscoringlist,x)+1))
    -- updatestage
  end repeat
end if

```

end

```
on chartCoal
  global
  gchartlist, thegame, goldscoringlevel, gsfx, scoringlevel, highlevel, minlist, maxlist, spritelis
  t, monsterlist,
  ballchart, ballmaster, goldscoringlist, gscoringlist, newscorelist, whatgame, gbigchartlist, cha
  rtpro, specialsound
  set specialsound = ""
  set ballchart = [{"c_1_0", "c_1_1"}, {"c_7_0", "c_7_1", "c_7_2", "c_7_3", "c_7_4"}]
  set ballmaster = [1,1,1,1,1,1,2]
  set whatgame = "6"
  set maxlist = [1,1,1,1,1,1,4]
  set minlist = [0,0,0,0,0,0,0]
  set spritelist = [27,28,29,30,31,32,33]
  put getat(gbigchartlist,6) into gscoringlist
  set Coallist = list()
  put getat(gchartlist,6) into Coallist
  put getat(Coallist,1) into Coalscoringlevel
  put getat(Coallist,2) into Coalhighlevel
  if Coalhighlevel = 0 then
    set Coalhighlevel = 1
  end if

  set levelCoallist =
  [[0,10],[11,20],[21,34],[35,40],[41,52],[53,56],[57,63],[64,66],[67,72],[73,74],[75,76]]

  put count(levelCoallist) into listNum
  repeat with y = 1 to listNum
    put getat(levelCoallist,y) into checklevel
    put getat(checklevel,1) into d
    put getat(checklevel,2) into e
    if (Coalscoringlevel >= d) and (Coalscoringlevel <= e) then
      set scoringlevel = y
    end if
    if (Coalhighlevel >= d) and (Coalhighlevel <= e) then
      set Coalhighlevel = y
    end if

    if thegame = "6" then
      if (goldscoringlevel >= d) and (goldscoringlevel <= e) then
        set goldscoringlevel = y
      end if
    end if
  end repeat

  put "Coalscoringlevel = "& scoringlevel
  put "Coalhighlevel = "& Coalhighlevel
```



```
if scoringlevel > 11 then
    set scoringlevel = 11
end if
```

```
if Coalhighlevel > 11 then
    set Coalhighlevel = 11
end if
```

```
--
--
--
```

```
GivemeBalls
```

```
if chartpro = "0" then
    set CoalVspritelist = list(1000,309,309,275,275,241,241,207,207,173,173)
    set CoalHspritelist = list(1000,142,172,142,172,142,172,142,172,142,172)
    put getat(CoalVspritelist,Coalhighlevel) into y
    put getat(CoalHspritelist,Coalhighlevel) into x
    set the locV of sprite 38 to y
    set the locH of sprite 38 to x
    set the visible of sprite 38 to true
    -- updatestage
else
    set the loc of sprite 38 to point(1000,1000)
    set the visible of sprite 38 to false
end if
```

```
end
```

Score Script5

```
on exitFrame
    go to the frame
end
```

Script of Cast Member6

```
on mouseUp
    global gRecordKeeper
    set newData = 0
    put the text of member "name" & Return into newData
    put the text of member "age" & Return after newData
    put the text of member "nickname" & Return after newData
    storeData gRecordKeeper, "user1", newData, true
```

```
end
```

Script of Cast Member7

```
) on mouseUp
    global gRecordKeeper
    restoreData gRecordKeeper, "user1", 2, "name"
    restoreData gRecordKeeper, "user1", 3, "age"
    restoreData gRecordKeeper, "user1", 4, "nickname"

end

--RestoreData me, WhichCast, WhichLine, whichmember
```

Script of Cast Member11

```
on mouseUp
    go to movie "dataview"
end
```

Script of Cast Member12

```
on mouseUp
    global gtheName
    put field "UserName" into theName
    set gtheName = theName
    go to movie "rappers"
end
```

Script of Cast Member13

```
on mouseUp
    global gtheName
    put field "UserName" into theName
    set gtheName = theName
    go to movie "c.c coal.dir"

end
```

Score Script14

Script of Cast Member15

Score Script16

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "3" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to frame "intro" of movie "Rhyme8"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
--end
```

```
on mouseUp
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "3" into theGame
    go to frame "intro" of movie "Rhyme8"
```

```
end
```

Script of Cast Member17

```
on mouseUp
    global gRecordKeeper
    put field "userName" into theName
    addUser(gRecordKeeper, theName)
end
```

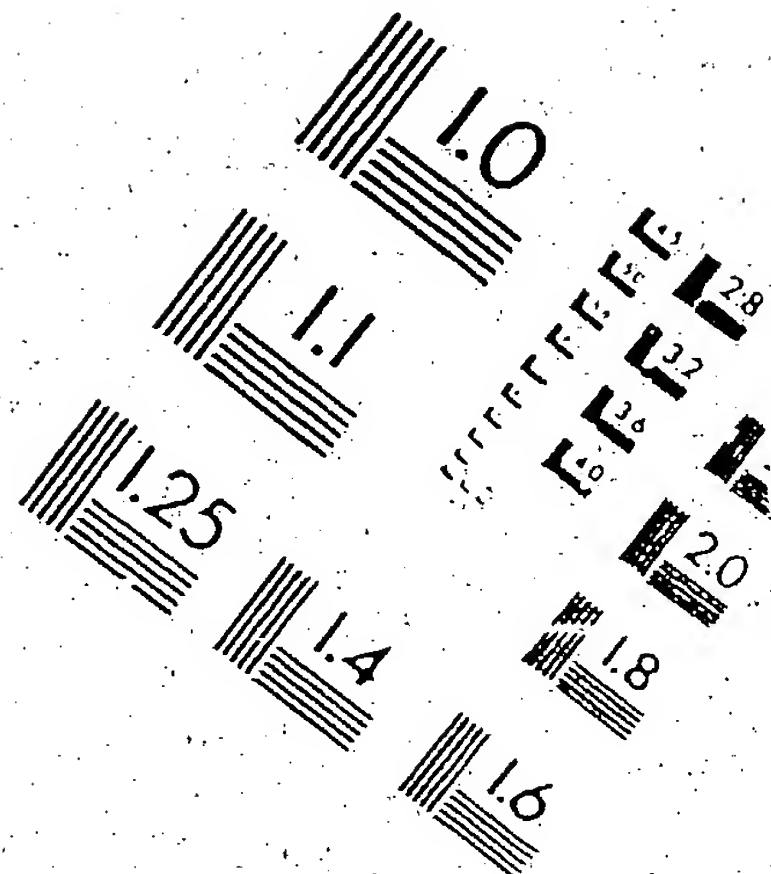
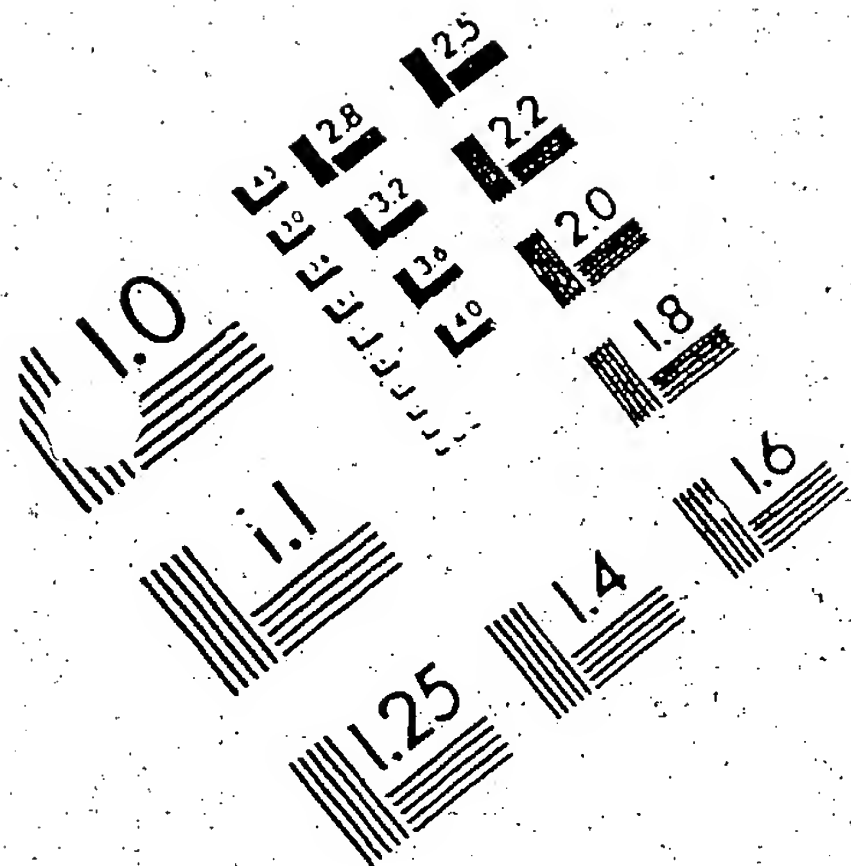
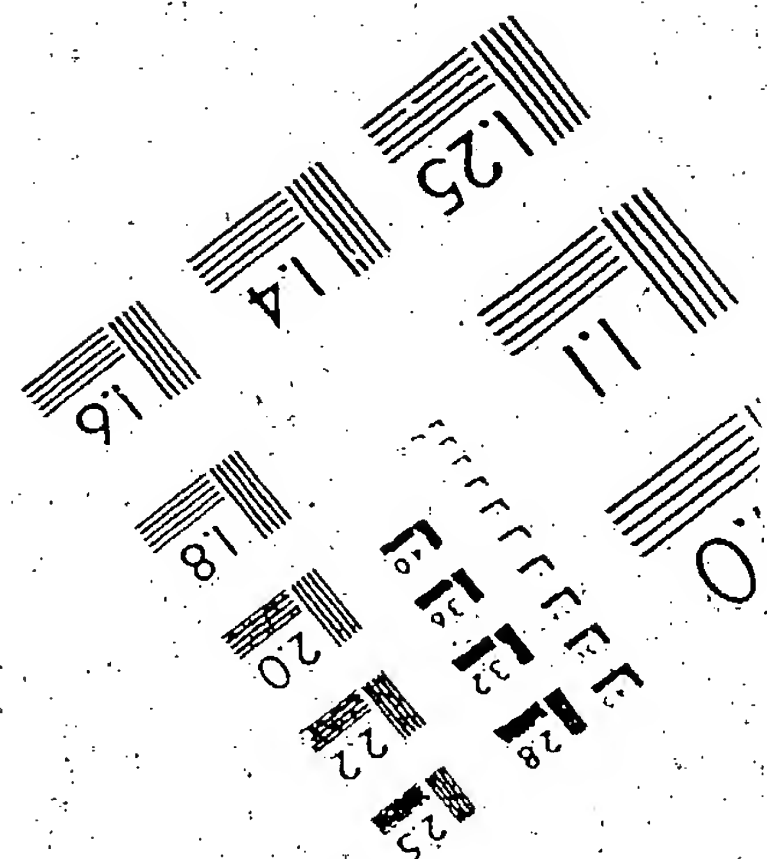
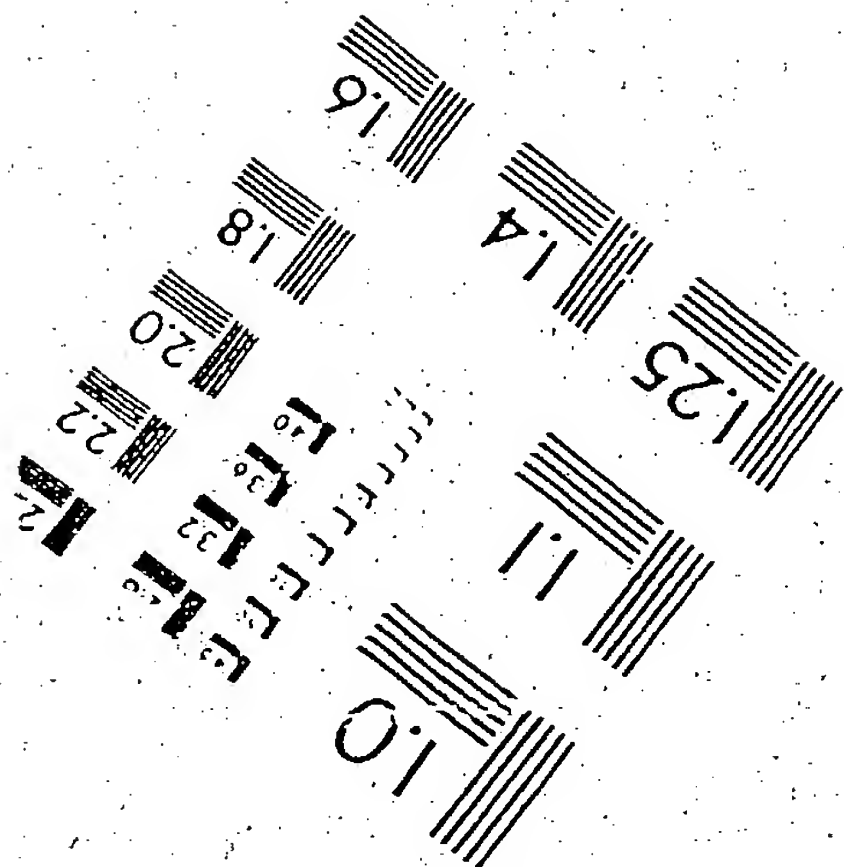
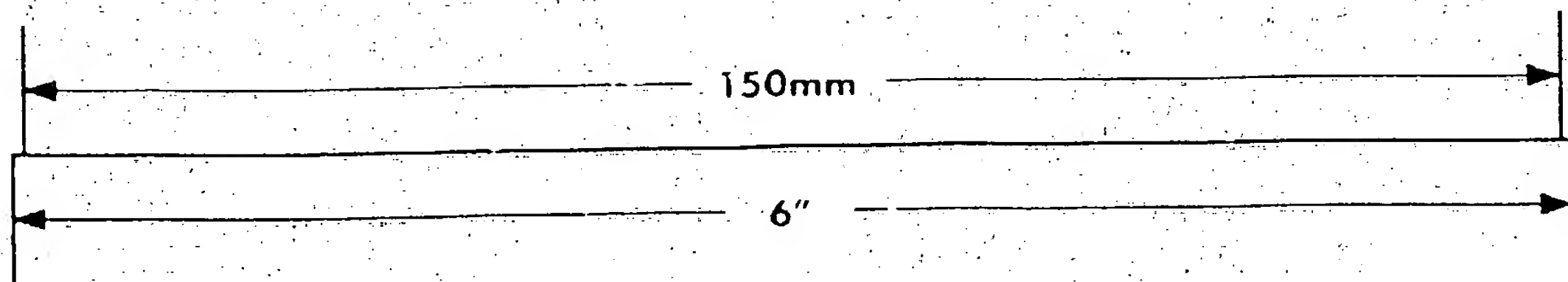
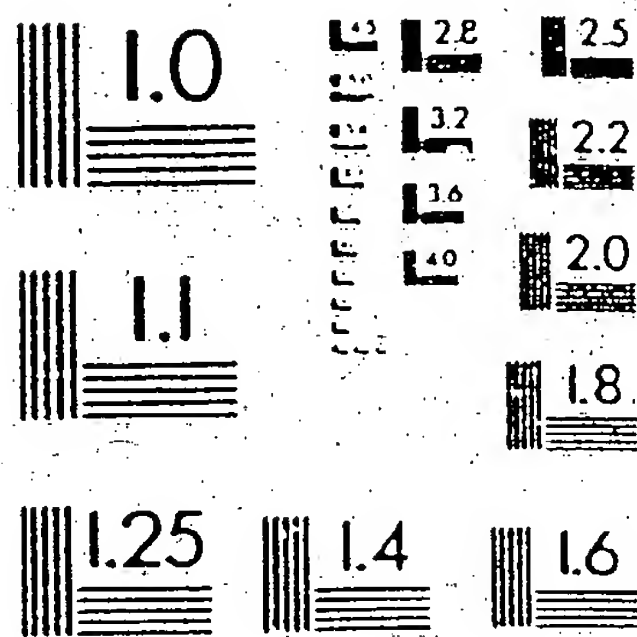


IMAGE EVALUATION TEST TARGET QA-3



APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone 716/482-0300
Fax 716-289-5989

Script of Cast Member18

```
on mouseUp
    global gRecordKeeper
    put field "UserName" into theName
    put field "GameNum" into theGame
    setUpRound(gRecordKeeper, theName, Value(theGame))
end
```

Script of Cast Member20

```
on mouseUp
    global gRecordKeeper
    set Numplays = item 1 of field "Scores"
    set NumRight = item 2 of field "scores"
    addToScore(gRecordKeeper, Value(Numplays), Value(NumRight))
end
```

Script of Cast Member21

```
on mouseUp
    global gRecordKeeper
    put field "Level" into level
    setScoringLevel (gRecordKeeper, level)
end
```

Script of Cast Member22

```
on mouseUp
    global gRecordKeeper
    put field "Level" into level
    setSavedLevel (gRecordKeeper, level)
end
```

Script of Cast Member23

```
on mouseUp
    global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName, gnextgame
    put field "UserName" into theName
    set gtheName = theName
    put field "GameNum" into theGame
    setUpRound(gRecordKeeper, theName, Value(theGame))
    go to frame "intro" of movie "Rhyme8.dir"
    set gscoringlevel = the result
    set gsavedlevel = gscoringlevel
    set gnextgame = gscoringlevel
end
```

Script of Cast Member24

```
on mouseUp
    global gtheName
    put field "UserName" into theName
    set gtheName = theName
    go to movie "Eggs8.dir"
end
```

Score Script25

```
on mouseUp
    put "Player 1" into field "username"
end
```

Score Script26

```
on mouseUp
    put "Player 2" into field "username"
end
```

Score Script27

```
on mouseUp  
    put "Player 3" into field "username"  
end
```

Score Script28

```
on mouseUp  
    put "Player 4" into field "username"  
end
```

Score Script29

```
on mouseUp  
    put "Player 5" into field "username"  
end
```

Score Script30

```
on mouseUp  
    put "Player 6" into field "username"  
end
```

Score Script31

```
--on mouseUp
--  global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
--  put field "UserName" into theName
--  set gtheName = theName
--  put "6" into theGame
--  setUpRound(gRecordKeeper, theName, Value(theGame))
--  go to movie "coal8"
--  set gscoringlevel = the result
--  set gsavedlevel = gscoringlevel
--end
```

```
--
on mouseUp
  global gtheName, theGame
  put field "UserName" into theName
  set gtheName = theName
  put "6" into theGame
  go to movie "coal8"
```

end

Score Script32

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "4" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to movie "rappers8"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
--end
```

```
on mouseUp
  global gtheName, theGame
  put field "UserName" into theName
  set gtheName = theName
  put "4" into theGame
  go to movie "rappers8"
end
```

Score Script33

```
--on mouseUp
--  global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
--  put field "UserName" into theName
--  set gtheName = theName
--  put "2" into theGame
--  setUpRound(gRecordKeeper, theName, Value(theGame))
--  go to movie "Eggs8"
--  set gscoringlevel = the result
--  set gsavedlevel = gscoringlevel
--end
```

```
on mouseUp
  global gtheName, theGame
  put field "UserName" into theName
  set gtheName = theName
  put "2" into theGame
  go to movie "Eggs8"
end
```

Score Script35

```
--on mouseUp
--  global gRecordKeeper,gscoringlevel,gsavedlevel,gtheName
--  put field "UserName" into theName
--  set gtheName = theName
--  put "1" into theGame
--  setUpRound(gRecordKeeper, theName, Value(theGame))
--  go to movie "katy8"
--  set gscoringlevel = the result
--  set gsavedlevel= gscoringlevel
--end
```

```
on mouseUp
  global gtheName,theGame
  put field "UserName" into theName
  set gtheName = theName
  put "1" into theGame
  go to movie "Katy8"
```

```
end
```

Score Script36

```
--on mouseUp
-- global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
-- put field "UserName" into theName
-- set gtheName = theName
-- put "5" into theGame
-- setUpRound(gRecordKeeper, theName, Value(theGame))
-- go to frame "mac"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
-- go to movie "Karloon8"
-- set gscoringlevel = the result
-- set gsavedlevel= gscoringlevel
--
--end
```

```
on mouseUp
    global gtheName, theGame
    put field "UserName" into theName
    set gtheName = theName
    put "5" into theGame
    go to movie "Karloon8"
```

```
end
```

Score Script37

```
on exitFrame
  global chartpro
  repeat with x = 4 to 33
    puppetsprite x, true
  end repeat
  repeat with x = 37 to 42
    puppetsprite x, true
    if chartpro = "1" then
      set the visible of sprite x to false
    end if
  end repeat

  showmeballs

  cursor -1
end
```

Score Script38

```
on exitFrame
  go to the frame
end
```

Score Script39

```
on exitFrame
  go to frame 5
end
```

Score Script47

```
on mouseUp
  cursor 4
  repeat with x = 1 to 48
    puppetsprite x, false
  end repeat

  go to movie "datatest"
end
```

Script of Cast Member48

Score Script50

```
--on mouseUp
--  global gRecordKeeper, gscoringlevel, gsavedlevel, gtheName
--  put field "UserName" into theName
--  set gtheName = theName
--  put "3" into theGame
--  setUpRound(gRecordKeeper, theName, Value(theGame))
--  go to frame "intro" of movie "Rhyme8"
--  set gscoringlevel = the result
--  set gsavedlevel = gscoringlevel
--end

on mouseUp
  global gtheName, theGame
  put field "UserName" into theName
  set gtheName = theName
  put "3" into theGame
  -- go to frame "black"
  go to movie "Rhyme8"

end
```

Score Script52

```
on mousedown
  buttonDownhandler
end
on mouseup
  buttonUPhandler
end
```

Score Script72

```
on mouseUp
  cursor 4
  set doc = 0
  set doc = new(xtra "printomatic")

  if not objectP(doc) then
    alert "there is a problem with printing"
    cursor -1
    exit
  end if

  reset doc
  setLandscapeMode (doc,true)
  setMargins(doc,rect(72,36,72,36))

  append doc, member "fake" of castLib 1, TRUE
  drawStagePicture doc, Point(0,0), Rect(0,0,640,480), TRUE

  print doc
  cursor -1
end
```

Score Script75

```
on mouseUp
  global spritelist

  if the machinetype = 256 and the colordepth > 8 then
    alert "Progress Chart is unable to print in Thousands(16 bit) or Millions(24 bit,32
    bit) of Colors, Please set your monitor to 256 Colors in the Monitors Control Panel."
    exit

  end if
  cursor 4
  set doc = 0
  set doc = new(xtra "printomatic")

  if not objectP(doc) then
    alert "there is a problem with printing"
    cursor -1
    exit
  end if

  reset doc
  setLandscapeMode (doc,true)
  setMargins(doc,rect(72,36,72,36))

  newPage doc
  -- append doc, member "fake" of castLib 1, TRUE
  drawStagePicture doc, Point(0,0), Rect(0,0,640,480), TRUE
  append doc, the date

  -- set spriteRect = the rect of member "print back"
  -- newFrame Doc, spriteRect, false
  -- append doc, member "print back", false
  --
  -- set spriteRect = the rect of sprite 17
  -- -- set spriteRect = rect(54, 18, 262, 32)
  -- newFrame Doc, spriteRect, false
  -- settextsize doc,14
  -- settextjust doc, "centered"
  -- append doc, member "UserName", false
  --
  --
  -- set spritelist = list()
  -- set spritelist = list(3,4,5,6,7,8,10,11,12,13,14,15,21,22,23,24,25,26)
  -- put count(spritelist) into countsprite
  -- repeat with y = 1 to countsprite
  --
  --   put getat(spritelist,y) into currsprite
  --   put currsprite
  --   -- set memberName = the Name of The Member of sprite currsprite
  --   --4 put "name = " & memberName
  --   set spriteRect = the rect of sprite currsprite
  --   newFrame Doc, spriteRect, false
  --   append doc, sprite currsprite, false
  --   -- append doc, member memberName, false
  -- end repeat
  -- -- append doc, member "ccal up",false

  setDocumentname (doc, "Earobics Progress Chart")
  print doc
```

```
cursor -1  
end
```

Score Script98

```
on mouseUp  
  if the machinetype = 256 and the colordepth > 8 then  
    alert "Progress Chart is unable to print in Thousands(16 bit) or Millions(24 bit,32  
bit) of Colors. Please set your monitor to 256 Colors in the Monitors Control Panel."  
    exit  
  end if  
  cursor 4  
  
  printfrom (the frame), (the frame), 50  
  cursor -1  
end
```

Score Script99

```
on mouseUp
  global gDataSavePath
  -- get coordinates of capture rectangle
  set left = 0
  set top = 0
  set right = 640
  set bottom = 480

  -- get filename argument
  set filename = "temp.bmp" -- string specified

  set retVal to StageToFile(left, top, right, bottom, filename)
  -- ShowRetVals(retVal, GetMessage(retVal))
  updateStage

  cursor 4
  set doc = 0
  set doc = new(xtra "printomatic")

  if not objectP(doc) then
    alert "there is a problem with printing"
    cursor -1
    exit
  end if

  reset doc
  setLandscapeMode (doc, true)
  setMargins(doc, rect(72, 36, 72, 36))

  newPage doc
  appendFile doc, gDataSavePath & "temp.bmp", TRUE

  append doc, the date
  setDocumentname (doc, "Earobics Progress Chart")
  print doc

  cursor -1
end
```

Score Script100

```
on mouseUp
  global gDataSavePath
  -- get coordinates of capture rectangle
  set left = 0
  set top = 0
  set right = 640
  set bottom = 480

  -- get filename argument
  set filename = "temp.bmp" -- string specified

  set retVal to StageToFile(left, top, right, bottom, gDataSavePath & "temp.bmp")
  -- ShowRetVals(retVal, GetMessage(retVal))
  updateStage
  -- starttimer
  -- repeat while the timer < 60
  -- end repeat

  starttimer
  repeat while fileExists(gdatasavepath & "temp.bmp") <> 0
    if the timer > 5*60 then
      alert "There is a problem with printing"
      cursor -1
      exit
    end if
  end repeat

  cursor 4
  set doc = 0
  set doc = new(xtra "printomatic")

  if not objectP(doc) then
    alert "There is a problem with printing"
    cursor -1
    exit
  end if

  reset doc
  setLandscapeMode (doc, true)
  setMargins(doc, rect(72, 36, 72, 36))

  newPage doc
  appendFile doc, gDataSavePath & "temp.bmp", TRUE

  append doc, the date
  setDocumentname (doc, "Earobics Progress Chart")
  print doc

  cursor -1
end
```

Movie Script1,

on StartMovie

```
global gDataSavePath, gRecordKeeper, gRecordDisplay, GoodCD, gchartlist
cursor 4
set gchartlist = list()
set GoodCD = False

put the pathname into gdataSavePath

-- save this location to a global so
-- that the "dataCollector" object can find the external cast "data.cst",
-- link to it and use it to store info from the movies on the locked volume.

set gRecordKeeper = 0
set gRecordKeeper = new(script "RecordKeeper")
-- set gRecordDisplay = 0
-- set gRecordDisplay = new(script "recordDisplay")

-- birth data collecting object
```

end

Parent Script2:recordKeeper

```
-- version 2/3/98
-- revised checkCastSizes handler
-- to continue trimming castmember until the size is
-- under 30500. Put a "repeat while" loop in to accomplish this
-- keeps out unlike possibility that on trim of record woul
-- not be enough --
-- Installed preference storage handlers
-- implement delete player stuffs(2/2/98)
property currentUser, currentGame, SavedLevel, ScoringLevel, HighLevel, proStatus,
CastLibName, newHigh, myHandlers, pObLevelTracker

on x-----Public Handlers -----
  -- I'm a separator
end

on AddUser me, UserName -- (function)
  -- pub.
  -- adds user name to member 1 of castLib "records.cst"
  -- here is where users are added and where the object
  -- checks to see if name is already on list or if more than
  -- 25 names are on the list. Does not limit length of name
  -- or any other quality of the name. Commas in the name would
  -- screw up my score reporting so they should be filtered out
  -- before giving the name to the recordKeeper
  -- Returns 1 if name was successfully added, Returns 0 if not

  case (proStatus) of
    0:set MaxUsers = 4
    1:set MaxUsers = 26
    2:set MaxUsers = 26
  end case

  openRecords me
  set users = GetNumberOfUsers(me)
  -- put the number of lines in field 1 of castLib castLibName into users
  if users >= MaxUsers then
    alert "Sorry, you have"&& MaxUsers&& "users already. You can not add any more users
to your records."
    closeRecords me
    return 0
  end if
  repeat with x = 1 to users
    if line x of field 1 of castLib castLibName = userName then
      alert "The name"&&Quote&userName&Quote&& "is already in use. Please choose a
different name. Thank you."
      closeRecords me
      return 0
    end if
  end repeat
  put userName & return after field 1 of castLib castLibName
  setUpNewUserMembers (me,userName) -- name members for new user and init those members
  save castlib castLibName
  closeRecords me
  return 1
end

on GetUserNames me -- (function)
```

```

-- Pub.
-- this function returns the names of users from cast 1 of
-- castLib "records.cst" They are returned as strings separated by
-- "returns" just as they are in the member. Be sure to
-- call as a function ie:
-- "put getUserNames(gRecordKeeper) into someVariable"
openRecords me
put field 1 of castLib castLibName into userList
CloseRecords me
put userList
return userList
end

on DeleteUser me, UserName
-- deletes user from player list and
-- takes name off of all records
-- though doesn't delete the records
set x = checkAndConvertUserName(me, userName, 1) -- lineNumber of player
if x = 0 then -- name not in list!
    alert "That name is not in the list."
    return 0
end if
openRecords me
delete line x of field "userNames" of castLib castLibName
repeat with y = 1 to 7 -- delete member names
    set memName = userName & y
    if the number of member memName <> -1 then -- watch for no prefs yet in "7"
        erase member memName of castLib castLibName
    end if
end repeat
saveRecords me
closeRecords me
end

on setUpRound me, whichUser, whichGame -- (function)
-- Pub.
-- This is the prime handler to be called at beginning of
-- any round of play. The game must tell "recordkeeper" whichUser and whichGame.
-- "WhichGame" must be an integer from 1 to 6
-- "WhichUser" is either string of user's name or an integer
-- representing the line on which user name is stored in
-- member 1 of castLib "Records.cst"
-- Call sets up a new game round for scoring.
-- ScoringLevel will be retrieved and set from
-- the saved level for that user and that game.
-- (item 1 of line 1 of member "currentUser&whichGame&")
-- It will also return the "scoringLevel" that it sets.
-- Use "the result" to get it and store in a variable.
-- for game 2 (eggBasket) returns a list with scoring level as
-- item 1 of the list and "bounceNum" as item 2 of the list
-- if the level of that game is over 30 (CV)
-- New(12/24/97) -- also reports scoring level to pObLevelTracker
-- to use for evaluating returned scores

setUser (me, whichUser)
setCurrentGame (me, whichGame)
openRecords me

```

```

set PrefCastMember = whichUser&"7"
if the number of member PrefCastMember of castLib castLibname = -1 then
    -- no prefs yet so build them
    makeNewPrefCast me, whichUser
    saveRecords me
end if

set thisRecord = currentUser&whichGame
set whichLevel = item 1 of line 1 of field thisRecord of castLib castLibname
set HighLevel = item 3 of line 1 of field thisRecord of castLib castLibname
if currentGame = 2 and whichLevel > 30 then
    set BounceNum = value(item 4 of line 1 of field thisRecord of castLib castLibname)
end if

if HighLevel = "" then set HighLevel = 1
set scoringLevel (me, whichLevel)
set savedLevel = whichLevel

put "current game =" && currentGame
put "Scoring level =" && scoringLevel
put "highLevel =" && HighLevel
setOpeningLevel(pObLevelTracker, scoringLevel) -- report to levelTracker
if CurrentGame = 2 and ScoringLevel > 30 then
    put "BounceNum =" && bounceNum
    set info = []
    setat (info, 1, scoringLevel)
    setat (info, 2, bounceNum)
    saveRecords me -- save report to level tracker
    closeRecords me
    return info
else
    saveRecords me -- save report to level tracker
    closeRecords me
    return scoringLevel
end if
end

on GetScoringLevel me -- (function)
    -- pub.
    -- returns the current "scoringLevel" property

    if voidP(scoringLevel) then exit
    put scoringLevel
    return scoringLevel
end

on SaveRoundScores me, roundlist, levelToSave
    -- pub.
    -- Reports scores from one round of play.
    -- Takes a list of the round's levels and scores.
    -- in case of game 2 it is a list of property lists (EggBasket)
    -- with the property being the level.
    -- otherwise it is a list of linear lists with the first value
    -- being the level. also takes a parameter for savedLevel.
    -- doesn't save scores till everything is reported.
    openrecords me
    set LevelsPlayedList = [] -- keep list of levelsPlayed for pObLevelTracker
    if currentGame <> 2 then
        repeat with thisList in roundlist
            put getat(thisList, 1) into thisLevel
            append LevelsPlayedList, thisLevel
        end repeat
    end if
end

```

```

    setScoringLevel me, thislevel
    addtoScore me, getat(thisList,2),getat(thisList,3)
end repeat
else -- special case for EggBasket
    set levelNums = count(roundlist)
    repeat with x = 1 to levelNums
        put getPropat(roundlist, x) into thislevel
        append LevelsPlayedList, thislevel
        setscoringlevel me, thislevel
        put getaProp(roundlist,thislevel) into scores
        addtoscore me, getat(scores,1), getat(scores,2), getat(scores,3), getat(scores,4)
    end repeat
    CheckEggBasketBounceNum me
end if
setsavedLevel me, levelToSave

reportLevelData pObLevelTracker, LevelToSave, currentGame, currentUser

saverecords me
closeRecords me
end

```

```

on loadchart me, whichuser -- handler to generate a chartlist for progress
-- (function)
if voidP(whichUser) then set whichUser = currentUser
openrecords me
set chartlist = list()
repeat with x = 1 to 6
    set thisrecord = whichUser&x
    set whichLevel = item 1 of line 1 of field thisRecord of castLib castLibname
    set HighLevel = item 3 of line 1 of field thisRecord of castLib castLibname

    set templist= list()
    append templist,value(whichLevel)
    append templist, value(HighLevel)
    append chartlist,templist
end repeat
closerecords me
return chartlist
end

```

```

on getHighLevel me, whichUser, whichGame
-- Pub.
-- (function)
-- returns high level, stored in
-- item 3 of line 1 of field "whichUser"&"WhichGame"

if voidP(whichGame) then set whichGame = currentGame
if voidP(whichUser) then set whichUser = currentUser

if stringP(whichUser) then set whichUser = checkandConvertuserName (me,whichUser)
if whichUser = 0 then
    alert "This Name is not in the records. Check spelling or add this name to your
records."
    abort

```



```

end if
if integerP(whichUser) then set whichUser = checkandConvertuserName (me,whichUser)
if whichUser = 0 then
    alert "Check your data. There is no user name at that number"
    abort
end if

if not integerP(whichgame) or whichGame < 1 or whichGame > 7 then
    alert "Please indentify the game with an integer from 1 to 7."
    abort
end if

openRecords me
put whichUser&whichGame into whichRecord
put item 3 of line 1 of field whichRecord of castLib castLibname into highLevel
closeRecords me
return highlevel
end

on setNameNumber me, entername, enternumber
    -- jim's handler to init the game
    openrecords me
    put entername into line 1 of field 4 of castlib castlibname
    put enternumber into line 2 of field 4 of castlib castlibname
    saveRecords me
    closeRecords me
end

on getNameNumber me
    -- jim's handler to see if game has been initied
    openrecords me
    put line 1 of field 4 of castlib castlibname into entername
    put line 2 of field 4 of castlib castlibname into enternumber
    return enternumber
    closeRecords me
end

on setUserPassword me, password
    --12/30/97
    openrecords me
    put password into line 3 of field 4 of castlib castlibname
    saveRecords me
    closeRecords me
end

on getUserPassword me
    -- 12/30/97
    openrecords me
    put line 3 of field 4 of castlib castlibname into password
    return password
    closeRecords me
end

on setProStatus me, WhichIsIt
    -- Pub.
    -- sets professional status of this usage and writes
    -- status to member 2 of "Records.Cst" pass 1 if pro and
    -- pass 0 if not pro.

```

```

openrecords me
put whichIsit into line 3 of field 2 of castlib castlibname
saveRecords me
closeRecords me
set proStatus = whichisit
end

on getProStatus me -- (function)
-- pub.
-- retrieves ProfessionalStatus from member 2 of castLib "records.cst".
-- Returns the item as a string
-- reads line 3 of field 2 of castLib "records.cst"
-- if "" is there, it defaults to proStatus of 0 (home use)
openrecords me
put line 3 of field 2 of castlib castlibname into whichisit
if whichisit = "" then set whichisit = 0
set proStatus = whichisit
return whichisit
closeRecords me
end

on setUserPrefs me, whichUser, prefList
-- set up a seventh cast member for whichUser named whichUser&"7"
-- takes a list of 6 sublists, each of four items for the four
-- preferences to be stored for each game. stores the lists on
-- six separate lines, line 1 for game 1 line two for game two etc.
set PrefCast = whichUser&"7"
openRecords me
if the number of member PrefCast of castLib castlibname = -1 then
    makeNewPrefCast me, whichUser
end if
repeat with x = 1 to 6
    put getAt(prefList, x) into gameList
    repeat with y = 1 to 4
        if y <= 3 then -- first three are toggles
            put getAT(gameList, y) into item y of line x of member PrefCast of castlib
castlibname
        else -- fourth sets new level to start at
            if getAT(gameList, y) = 0 then next repeat -- no positive integer, no change
            set thisRecord = whichUser&x
            if x = 2 then -- egg basket so reset bounceNum also
                put 0 into item 4 of line 1 of member thisRecord of castLib castLibName
            end if
            put getAT(gameList, y) into item 1 of line 1 of member thisRecord of castLib
castLibName
            -- I purposely choose not to call setSavedLeved here so as not to
            -- change High score until player plays at new high level
            adjustLevelListForChangedLevel pObLevelTracker, whichUser, x, getAT(gameList, y)
            -- above, we need to wipe out all completed levels for the new category which
            -- the user is going into, if any completed levels are there.
        end if
    end repeat
end repeat
saveRecords me
closeRecords me
end

on getUserPrefs me, whichUser
-- returns list of 6 sublists
-- each is games prefs.item 1 of list = game 1 prefs etc.
set PrefCast = whichUser&"7"

```

```

openRecords me
if the number of member PrefCast of castLib castlibname = -1 then
    makeNewPrefCast me, whichUser
    SaveRecords me
end if
put field PrefCast of castLib castlibname into prefs
set PrefList = []
repeat with game = 1 to 6
    set tempList = []
    repeat with y = 1 to 4
        if y mod(4) = 0 then -- next to get current level and convert
            set thisPref = getLevelNum(pObLevelTracker, game, whichUser)
        else
            set thisPref = value(item y of line game of prefs)
        end if
        append tempList, thisPref
    end repeat
    append prefList, tempList
end repeat
closeRecords me
return prefList
end

on getUserSkipGameList me, whichUser
    -- returns six item list of booleans
    -- 1 means skip that game, 0 means play that game
    openrecords me
    set PrefCast = whichUser&"7"
    if the number of member PrefCast of castLib castlibname = -1 then
        makeNewPrefCast me, whichUser
        SaveRecords me
        closeRecords me
        return [0,0,0,0,0,0] -- no prefs so return "no skip" list
    end if
    set skipList = []
    put field PrefCast of castLib castLibName into prefs
    repeat with x = 1 to 6
        append skipList, value(item 1 of line x of prefs) -- skipped games stored here
    end repeat
    closeRecords me
    return skipList
end

on getGamePrefs me, whichUser, whichGame
    -- returns two item list of booleans
    -- first: whether to disable "replay" button, 1 = disable, 0 = don't disable
    -- second: whether to disable "YesOrNo" prompt at end, 1 = disable, 0 = don't disable
    openrecords me
    set PrefCast = whichUser&"7"
    if the number of member PrefCast of castLib castlibname = -1 then
        makeNewPrefCast me, whichUser
        saveRecords me
    end if
    set gamePrefList = []
    put field PrefCast of castLib castlibname into prefs
    append gamePrefList, value(item 2 of line whichGame of prefs) -- disable replay button?
    append gamePrefList, value(item 3 of line whichGame of prefs) -- disable yesOrNo?
    closeRecords
    return gamePrefList
end

```



```

on getGameLevelLists me , whichuser
-- pass request to level tracker
-- returns list used by progress chart and
-- to display check marks in prefs screen.
return getGameLevelLists (pObLevelTracker , whichuser)
end

---jim changed
on new me
-- Pub.
global gCastLibName
set castLibName = "Records.cst" -- cast where records are stored --jim changed
set gCastLibName = castLibName
set myHandlers = 0
set myHandlers = GetMyHandlers(me)
initProStatus me -- read proStatus from member 2 of cast "Records.cst"
set pObLevelTracker = new(script "completedLevelTracker",castLibName)
checkCastSizes me -- call maintenance function to trim members over 30500 characters
if the name of member 10 of castLib castLibName <> "DefaultPrefs" then
-- need to install default cast member first time new version starts up
-- may act as a marker for other first time stuff
duplicate member "DefaultPrefs" of castLib 1, member 10 of castLib castLibName
save castlib castLibName
end if
return me
end

on xx-----Private Handlers-----
-- i'm a separator
end

on AddToScore me, howManyPlays, howManyRight, howManyPlays2, howManyRight2
-- Pub.
-- main handler for writing scores to "Records"
-- must be called after a new game is set up and user
-- etc, are all selected. Last two params are only used in
-- "BasketofEggs" game.

if VoidP(currentUser) or VoidP(scoringLevel) then
-- check that "SetUpRound" has been called to init values for
-- UserName and CurrentGame. If not alerts will come up.

alert " Make sure to use handler"&&QUOTE&"SetUpRound"&&QUOTE&"before using
handler"&&QUOTE&"AddToScore"&&QUOTE&". Please check your lingo."
abort
end if

if currentGame = 2 then
-- check to see if we are playing "basket o eggs" and if
-- so check that the correct number of parameters are being passed
-- also set a flag for adding the two extra params down the line
if (voidP(howManyPlays2) or voidP(howManyright2)) then
alert "Basket-of- Eggs takes two extra scores. please check your lingo"
abort
else
set EggBasket = true
end if
end if
end if

```

```

set thisRecord = currentUser&CurrentGame
put field thisRecord of castlib castLibname into record
-- puts everything into variable for faster access

put item 1 of line 2 of record into lastDate -- check date of last writing
if lastDate <> the date then -- first time writing to this game in this session
    put the date into item 1 of line 2 of record -- store today's date
    put the number of lines of record into item 2 of line 2 of record -- store starting
point
end if

put integer(item 2 of line 2 of record) into whichLine
put the number of lines of record into numLines

if (whichLine+1) > numLines then
    -- we haven't yet written on this date
    -- so we make a new record at next line
    -- and save new number of sessions
    put the date into item 1 of text
    put scoringLevel into item 2 of text
    put howmanyPlays into item 3 of text
    put howmanyRight into item 4 of text

    if EggBasket then -- basketofEggs!!
        put howmanyPlays2 into item 5 of text
        put howmanyRight2 into item 6 of text
    end if

    put text into line (whichLine+2) of record
    put integer(item 2 of line 1 of record) into numSessions -- get number of previous
sessions
    set numSessions = numSessions + 1 -- increment for today's session
    put numSessions into item 2 of line 1 of record -- store new value
    put record into field thisRecord of castlib castLibname
    exit
end if

-- if we get here there is a record already stored so we check
-- to see if it is at same level as scoring level
-- if it is we adjust the values of this line accordingly

repeat with x = (whichLine+1) to numLines
    if (item 1 of line x of record = the date) and (item 2 of line x of record =
scoringLevel) then
        set NumPlays = item 3 of line x of record
        set numRight = item 4 of line x of record
        set numPlays = numPlays + howmanyPlays
        set numRight = numRight + howManyRight
        put integer(numPlays) into item 3 of line x of record
        put integer(numRight) into item 4 of line x of record
        if eggBasket then
            set NumPlays = item 5 of line x of record
            set numRight = item 6 of line x of record
            set numPlays = numPlays + howmanyPlays2
            set numRight = numRight + howManyRight2
            put integer(numPlays) into item 5 of line x of record
            put integer(numRight) into item 6 of line x of record
        end if

        put record into field thisRecord of castlib castLibname
    end if
end repeat

```

```

        exit
    end if
end repeat

-- if we get here then there is no record for this level at this date
-- so we make one up
put the date into item 1 of text
put scoringLevel into item 2 of text
put howmanyPlays into item 3 of text
put howmanyRight into item 4 of text
if eggBasket then
    put howmanyPlays2 into item 5 of text
    put howmanyRight2 into item 6 of text
end if

put text into line (x) of record
put record into field thisRecord of castlib castLibName
exit

end

on setSavedLevel me, whichLevel
    -- pub.
    -- use to save level at which next game is to be played
    -- most time it is same as "scoringLevel" but in special
    -- case where user has jumped levels to a new "skill game"
    -- the rest of round is played at the old (scoringLevel) level
    -- but the next round will open to the new level. Easy huh?
    set WhichLevel = Value(WhichLevel)
    set thisRecord = currentUser&CurrentGame
    put item 3 of line 1 of field thisRecord of castLib castLibName into HiLevel
    put whichLevel into item 1 of line 1 of field thisRecord of castLib castLibName
    if value(whichLevel) > value(HiLevel) or HiLevel = "" then
        put whichLevel into item 3 of line 1 of field thisRecord of castLib castLibName
        if currentGame = 2 and scoringLevel > 30 then
            -- reset counter for EggBasket bounce function
            -- when the counter passes 5 we bounce the user to
            -- new sound family
            put 0 into item 4 of line 1 of field thisRecord of castlib castLibName
        end if
    end if
end if
set savedLevel = whichLevel
end

on setScoringLevel me, whichLevel
    -- pub./priv.
    -- use to tell recordKeeper at which level to write out current score.
    -- Also checks whichLevel against HighLevel (pulled out of item 3 of
    -- line 1 of the current record. If whichLevel is higher it
    -- replaces stored highlevel and property highlevel with whichLevel
    set WhichLevel = Value(WhichLevel)
    set scoringLevel to whichLevel
    if Value(whichLevel) > Value(highLevel) then
        set thisRecord = currentUser&CurrentGame
        put whichLevel into item 3 of line 1 of field thisRecord of castlib castLibName
        set HighLevel = whichLevel
        if currentGame = 2 and scoringLevel > 30 then
            -- reset counter for EggBasket bounce function
            -- when the counter passes 5 we bounce the user to
            -- new sound family
            put 0 into item 4 of line 1 of field thisRecord of castlib castLibName
        end if
    end if
end if

```

```

        set NewHigh = true
    end if
end if
end

on CheckEggBasketBounceNum me
    if scoringLevel < 31 then exit
    set thisRecord = currentUser&currentGame
    if NewHigh <> true then
        put value(item 4 of line 1 of field thisrecord of castLib castLibName) into numRounds
        set numRounds = numRounds + 1
        put numRounds into item 4 of line 1 of field thisrecord of castLib castLibName
    else
        set newHigh = false
    end if
end

on setUser me, user
    -- priv.
    -- takes either user's name (must be exact) and checks it against the
    -- list of users in cast member 1, or will take a number and retrieve
    -- the user name as line "user" of cast member 1. Also checks to see
    -- that there is a name at that line
    if stringP(user) then
        set user = checkandConvertUserName(me, User)
        if user = 0 then
            alert "This Name is not in the records. Check spelling or add this name to your
records."
            abort
        else
            set currentUser = user
            put "Current User =" && currentUser
            exit
        end if
    end if
    if integerP(user) then
        set user = checkandConvertUserName(me, User)
        if user = 0 then
            alert "Check your data. There is no user name at that number"
            abort
        else
            set currentUser = user
            put "Current User =" && currentUser
        end if
    end if
end

on checkAndConvertUserName me, whichName, ReturnLineNum -- function
    -- Priv.
    -- Takes a string and checks to see if it is in list of
    -- user names stored in field 1 of castLib "records.cst".
    -- if it is it returns the string, if not it returns 0
    -- if ReturnLineNum = 1, returns the lineNumber for deleting(2/2/98)
    -- Also will take an integer and check to see if there is a name
    -- at that line of field 1. If so it returns the name on that
    -- line, if not it returns 0.

    if not stringP(whichName) and not integerP(whichName) then
        alert "Expecting a string or an integer. Check your lingo."
        abort
    end if

```



```

if integerP(whichName) and whichName < 1 then
    alert "Expecting a positive integer. Check your Lingo."
    abort
end if

openRecords me
set userList = field 1 of castLib castLibName
closeRecords me
set user = 0

if stringP(whichName) then -- check to see if name is on the list
    set lastLine = the number of lines in userList
    repeat with x = 1 to lastLine
        if line x of userList = whichName then
            set user = whichName
            exit repeat
        end if
    end repeat
    if returnLineNum = 1 then -- need line in list
        if user = 0 then
            return user -- not in list
        else
            return x
        end if
    else -- just confirming name is in list
        return user
    end if
else -- must be integer so see if there is a name on that line
    if line whichName of userList = "" then
        return 0
    else
        set user = line whichName of userList
        return user
    end if
end if
end

on setUpNewUserMembers me, userName
    -- priv.
    -- sets up 6 castnames for each new user, one per game.
    -- Initializes those cast members with a line 1 of "1,0"
    -- "1" stands for savedLevel of game, always starts at level 1
    -- "0" stands for current number of sessions played of game
    -- Line 2 is initted with ",2". Item one will hold
    -- the date of latest play, item 2 is line to start writing at.
    set Num = (((the number of members of castLib castLibName -1) /10 )+1)*10)
    repeat with x = 1 to 6
        set newMemb = Num + x -- make new member number
        duplicate member "sampleCast" of castLib castLibName, member newMemb of castLib
        castLibName
        put "" into member newMemb of castLib castLibName
        set the name of member newMemb of castLib castLibName = userName&x
        put "1,0" into line 1 of member newMemb of castLib castLibName
        put ",2" into line 2 of member newMemb of castLib castLibName
    end repeat
    makeNewPrefCast (me, userName)
end

on makeNewPrefCast me, whichUser

```

```

-- sets up 7th cast member to hold preferences for user
-- feeds in default member as a template.
-- Make sure defaults are correct
set lastcast = whichUser&"6"
set newNum = (the memberNum of (member lastcast of castLib castLibName) ) + 1
duplicate member "DefaultPrefs" of castLib castLibName, member newNum of castLib
castLibName
set newName = whichUser&"7"
set the name of member newNum of castLib castLibName = newName

-- next build completed level lists from old data
set CompletedLevelLists = buildLevelLists(pObLevelTracker, whichUser)
repeat with x = 1 to 6
    put getAT(CompletedLevelLists, x) into line x + 6 of field newName of castLib
castLibName
end repeat
end

```

```

on setCurrentGame me, whichGame
-- priv.
--use to tell recordKeep which game is being played
-- expects integer from 1 to 6, not string of name of game
if not integerP(whichGame) or whichGame < 1 or whichGame > 6 then
    alert "Expecting an integer from 1 to 6. Please check your lingo"
    abort
end if
set currentGame = whichGame
end

```

```

on initProStatus me
-- priv.
-- use to initialize professional status on start-up
-- hence no "openRecords" command
put line 3 of field 2 of castlib castlibname into whichisit
if whichisit = "" then set whichisit = 0
set proStatus = whichisit
return whichisit
end

```

```

on GetMyHandlers me
-- Priv.
-- returns list of handlers to property variable
put value(word 2 of string(me)) into whichCast
put the scripttext of member, whichCast into text

put the number of lines in text into scriptLines
Put whichCast && "handlers"& return into Handlers

repeat with x = 1 to scriptLines
    if word 1 of line x of text = "on" then
        put line x of text into handlerName
        delete word 1 of handlerName
        put handlerName & return after handlers
    end if
end repeat
return handlers
end

```

```

on GetNumberOfUsers me
-- Priv.
-- walks the castmember "userNames" line by line
-- and counts the number of lines which hold a name
-- ie. are not just ""
put field "userNames" into temp
set y = the number of lines of temp
set NameCount = 0
repeat with x = 1 to y
    if line x of field "userNames" <> "" then set NameCount = NameCount + 1
end repeat
return NameCount
end

on checkCastSizes me
-- run through records cast and look all
-- members for number of chars (all field members that is)
-- if numChars is greater than 30000 (or whatever) then
-- call a handler to delete the earliest session in the record.
put "checking record sizes"
set n = the number of members of castLib castlibname
set fTrimmed = false
repeat with x = 4 to n -- first three members are not records.
    if the type of member x of castLib castlibname = #field then
        set numChars = the number of chars of field x of castLib castlibname
        if numChars > 30500 then
            repeat while the number of chars of field x of castLib castlibname > 30500
                put "Member "&x&" Contains"&&numChars&"characters. Deleting oldest session in
member."
                trimMember me ,x
            end repeat
            set fTrimmed = true
        end if
    end if
end repeat
if fTrimmed then
    save castlib castlibname
end if
put "done checking record sizes"
end

on trimMember me, whichMember
-- first find the first line after line 2 which has a date as first item
set n = the number of lines of field whichMember of castLib castlibname
repeat with y = 3 to n -- sessions start after line 2
    put item 1 of line y of field whichMember of castLib castlibname into testDate
    if testDate <> "" then -- look for date string
        exit repeat -- y will be the first line with a date.
    end if
end repeat

-- now loop through to count the lines which have the same date
put item 1 of line y of field whichMember of castLib castlibname into theDate
set endLine = y
repeat with x = (Y+1) to n
    if item 1 of line x of field whichMember of castLib castlibname = theDate then
        set endLine = endLine + 1
    else
        exit repeat
    end if
end if

```



```

end repeat

-- now delete the lines plus one more to keep empty lines from growing.
delete line y to endLine + 1 of field whichMember of castLib castlibname

-- finally decrement session counter to reflect deleted session
-- and adjust line count in case trimming occurs during
-- a session or between plays on same day. Causes recordKeeper to write
-- two lines of data for same level in that case.
set linesDeleted = ((endLine + 1) - y) + 1
put integer(item 2 of line 2 of field whichMember of castLib castlibname) into numLine
set NumLines = numLines - linesDeleted
put NumLines into item 2 of line 2 of field whichMember of castLib castlibname

put integer(item 2 of line 1 of field whichMember of castLib castlibname), into
numsessions
-- get number of previous sessions
set numsessions = numsessions - 1 -- decrement for today's session
if numsessions < 0 then set numsessions = 0
put numsessions into item 2 of line 1 of field whichMember of castLib castlibname
-- store new value
end

on openRecords me
-- priv.
-- opens castLib "Records.Cst" for read or write
global gDataSavePath
set the filename of castLib CastLibName = gDataSavePath & CastLibName
end

on SaveRecords me
-- Priv.
save castlib castLibName
end

On CloseRecords me
-- Priv.
set the filename of castLib CastLibName = the pathname & CastLibName
end

on xxx-----Testing Handlers-----
-- i'm a separator
nothing
end

on showHandlers me
-- Testing
-- puts list of handlers in message window
put myHandlers
end

on showProps me
-- testing
-- puts list of properties and their current values in message window
set PropNum = count(me)
repeat with x = 1 to PropNum
set prop = 0
set thisProp = getpropat(me, x)
if thisProp = #myHandlers then next repeat
put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop

```

```
put prop  
end repeat  
end
```

Parent Script3:CompletedLevelTracker

--1/4/98

property myHandlers, CastLibName, openningLevel

on x-----Public Handlers -----

-- I'm a separator

end

on buildLevelLists me, whichUser

-- for first time using version 2 with

-- old player. Looks at each game's high level and builds

-- a list of levels up to that. This works since version 1 always

-- ratchets up from level 1 on up. Must be called with records opened

-- by gRecordKeeper. Returns the list to the recordKeeper for saving

set LevelsList = []

repeat with x = 1 to 6

set tempList2 = []

set thisGameRecord = whichUser&x

set HighLevel = ""

put item 3 of line 1 of field thisGameRecord of castLib CastLibName into HighLevel

if highLevel = "" then

-- no play yet for game so just append the empty list

append LevelsList, tempList2

else

set highLevel = value(HighLevel)

put value(item 1 of line 1 of field thisGameRecord of castLib CastLibName) into

savedLevel

-- if game is maxed out savedLevel will be one unit higher than highLevel

set Level = max(savedLevel, HighLevel)

repeat with y = 1 to (Level-1)

-- highest level reached is always not yet completed, so next lower

-- level will be highest completed

append tempList2, y

end repeat

append LevelsList, tempList2

end if

end repeat

return LevelsList

end

on setOpenningLevel me, whichLevel

-- in order to track completed levels in version 2 we need to

-- know what level the round started at. When the round scores are reported

-- we can check the levels played and assume that any levels within the

-- openingLevel and (savedLevel - 1) have been completed.

-- this handler will be called only and everytime from the setUpRound handler in

-- gRecordKeeper and therefore should always be accurate

set openingLevel = whichLevel

end

on reportLevelData me, LevelToSave, whichGame, whichUser

-- here is main work of tracking and reporting

-- levels completed. This gets called from recordKeepers "saveRoundScores"

-- handler every time user's scores are reported. Params are

-- the saved level which is the next

-- level at which game will be played. comparing the openingLevel property

-- to the savedLevel will allow

-- us to determine which levels where actually completed.

```

-- if the saved level is less then the opening level then
-- the user dropped and we need to delete levels from the
-- completed levels list.
set UserPrefsCast = whichUser&7
set savedLevelsCompletedList = ~
value(line whichGame+6 of field UserPrefsCast of castLib castLibName)
-- get last list of completed levels
sort savedLevelsCompletedList
set x = (leveltoSave - openingLevel) -- what happened

set LevelsToAddList = []
set LevelsToDeleteList = []

if x < 0 then -- user dropped
  set x = abs(x)
  repeat with y = 0 to x
    -- create list of levels that user fell through
    append levelsToDeleteList, openingLevel - y
  end repeat
else if x > 0 then -- user rose
  repeat with y = 0 to (x - 1) -- saved level is never completed
    append LevelsToAddList, openingLevel + y
  end repeat
end if

if count(LevelsToAddList) then
  repeat with level in levelsToAddList
    if not getOne(savedLevelsCompletedList, level) then
      append savedLevelsCompletedList, level
    end if
  end repeat
end if

if count(levelsToDeleteList) then
  repeat with level in levelsToDeleteList
    if getOne(savedLevelsCompletedList, level) then
      deleteOne savedLevelsCompletedList, level
    end if
  end repeat
end if

set savedLevelsCompletedList = ~
correctLevelListForFallThru (me, savedLevelsCompletedList, levelsToDeleteList, whichGame)
-- need to see whether user fell through a category that was already
-- completed and if so delete those levels from the savedLevelsCompletedList
end repeat
end if

if not count(LevelsToAddList) and not count(levelsToDeleteList) then
  -- user started and stopped round at same level
  -- just to be sure we
  if getOne(savedLevelsCompletedList, leveltoSave) then
    deleteOne savedLevelsCompletedList, leveltoSave
  end if
end if

put "" into line whichGame+6 of field UserPrefsCast of castLib castLibName
put savedLevelsCompletedList into line whichGame+6 of field UserPrefsCast of castLib
castLibName
end

```

```

on correctLevelListForFallThru me, savedLevelsCompletedList, levelsToDeleteList, game
-- if user fell through a category line and if they had already

```

```

-- completed the catagory they fell out of we need to
-- delete the completed levels, so we check the deletedlevelist to see
-- if it contains a boundary level and if so we need to
-- delete all the levels in the catagory that the boundary level
-- came from
set gameListToCompare = getGameCatagoryList (me , game)
set tempList = []
repeat with level in levelsToDeleteList
  set testList = getAProp(gameListToCompare,level)
  if not VoidP(testList) then
    -- if the level is a property in the list
    -- we store the value associated with the level
    -- which is a list of 2 levels , we delete all levels between
    -- these two levels, inclusive.
    append tempList, getAProp(gameListToCompare,level)
  end if
end repeat
if count(tempList) then
  -- if list has items we need to take it appart
  -- and walk through all the lists in the list
  -- deleting those levels from are saved level list
  -- if they are there
  repeat with thisList in tempList
    repeat with thisLevel = getat(tempList,1) to getat(tempList,2)
      if getOne (savedLevelsCompletedList, thisLevel) then
        deleteOne savedLevelsCompletedList, thisLevel
      end if
    end repeat
  end repeat
end if
return savedLevelsCompletedList
end

on adjustLevelListForChangedLevel me, whichUser, whichGame, newStart
  -- if this is called it means teacher placed the student in a new
  -- catagory from the prefs screen. If so we need to wipeout all
  -- completed levels that may be in that catagory already so
  -- user starts with a clean slate there.
  set UserPrefsCast = whichUser&7
  put "UserPrefsCast = "&UserPrefsCast
  set savedLevelsCompletedList =-
  value(line whichGame+6 of field UserPrefsCast of castLib castLibName)
  -- get last list of completed levels
  sort savedLevelsCompletedList
  put "savedLevelsCompletedList = "&savedLevelsCompletedList

  set gameCatagoryList = getGameCatagoryList( me, whichGame)
  put "gameCatagoryList = "&gameCatagoryList
  -- get list of all levels in each catagory

  set newStart = convertLevel (me, newStart,whichGame)
  -- first convert new level to catagory number
  set levelsToClear = getAt(gameCatagoryList, newStart)
  put "levelsToClear = "&levelsToClear
  -- -- in order will give the levels for that catagory

  repeat with x = getat(levelsToClear,1) to getat(levelsToClear,2)
    if getOne(savedLevelsCompletedList,x) then deleteOne savedLevelsCompletedList,x
    -- if that level is there, wipe it out
  end repeat

```



```

    put savedLevelsCompletedList into line whichGame+6 of field UserPrefsCast of castLib
castLibName
    -- finally put the changed list back in the record. Calling handler will take
    -- care of saving and closing records

```

```

end

```

```

on getGameCatagoryList me , game

```

```

    -- stores list of intervals between catagories for the
    -- different games. Used by "correctLevelListForFallThru" routine
    -- to determine which levels to delete if user falls from one catagory
    -- to another, also used to determine which levels to delete
    -- if the teacher places child at a lower catagory with some
    -- or all levels already completed. In this case we delete all the
    -- levels for that catagory.

```

```

    case game of

```

```

        1: return [1:[1,12],13:[13,24],25:[25,28],29:[29,32],33:[33,44],45:[45,56]]

```

```

        2: return [1:[1,3], 4:[4,11],12:[12,20],21:[21,30],31:[31,58],59:[59,86],87:[87,114]]

```

```

        3: return [1:[1,5], 6:[6,11]]

```

```

        4: return [1:[1,6],7:[7,12],13:[13,14],15:[15,16]]

```

```

        5: return [1:[1,6],7:[7,18],19:[19,30],31:[31,38]]

```

```

        6: return

```

```

[1:[1,10],11:[11,20],21:[21,34],35:[35,40],41:[41,52],53:[53,56],57:[57,74]]

```

```

    end case

```

```

end

```

```

on getLevelNum me, game , whichUser

```

```

    -- called from recordKeeper when building prefsList to return to
    -- dataTest movie. Feed it the game and username and it
    -- returns an integer which tells the PrefScreen manager in dataTest
    -- which levels cast member (which line blue?) to put up

```

```

    set whichRecord = whichUser&game

```

```

    put item 1 of line 1 of field whichRecord of castLib castLibName into curLevel

```

```

    set whichBlueLine = convertLevel( me, curLevel,game)

```

```

    return whichBlueLine

```

```

end

```

```

on convertLevel me, curLevel,game

```

```

    -- knows which levels in each game correspond to

```

```

    -- which line is blue in the preferences screen.

```

```

    if game = 1 then

```

```

        case true of

```

```

            (curLevel>44):return 6

```

```

            (curLevel>32):return 5

```

```

            (curLevel>28):return 4

```

```

            (curLevel>24):return 3

```

```

            (curLevel>12):return 2

```

```

            (curLevel>= 0):return 1

```

```

        end case

```

```

    end if

```

```

    if game = 2 then

```

```

        case true of

```

```

            (curLevel>86):return 7

```

```

            (curLevel>58):return 6

```

```

            (curLevel>30):return 5

```

```

        (curLevel>20):return 4
        (curLevel>11):return 3
        (curLevel>3):return 2
        (curLevel>= 0):return 1
    end case
end if
if game = 3 then
    case true of
        (curLevel>5):return 2
        (curLevel>=0):return 1
    end case
end if
if game = 4 then
    case true of
        (curLevel>14):return 4
        (curLevel>12):return 3
        (curLevel>6):return 2
        (curLevel>=0):return 1
    end case
end if
if game = 5 then
    case true of
        (curLevel>30):return 4
        (curLevel>18):return 3
        (curLevel>6):return 2
        (curLevel>=0):return 1
    end case
end if
if game = 6 then
    case true of
        (curLevel>56):return 7
        (curLevel>52):return 6
        (curLevel>40):return 5
        (curLevel>34):return 4
        (curLevel>20):return 3
        (curLevel>10):return 2
        (curLevel>=0):return 1
    end case
end if
end

```

on getGameLevelLists me , whichuser

```

-- called for preferences and progressChart
-- returns list of 6 lists, each subList tracking Jan's chart of levels and chart balls
-- every position in the sublist tracks a pro-user settable point in the games levels.
-- the value of the list at that spot will be an integer that tells Jim how many balls
-- to fill in for that part of the progressChart. Also will tell me whether to put
-- a check mark on the prefs screen for that group of levels.

```

global gRecordKeeper

openRecords gRecordKeeper

set PrefsMember = whichUser&"7"

```

if the number of member PrefsMember of castLib CastLibName = -1 then

```

```

    makeNewPrefCast gRecordKeeper, whichUser

```

```

    saveRecords gRecordKeeper

```

```

end if

```

```

set GameLevelList = []

```

```

append GameLevelList, getCaterPillarLevels(me,whichuser)

```



```

append GameLevelList, getEggBasketLevels(me, whichuser)
append GameLevelList, getRhymeTimeLevels(me, whichuser)
append GameLevelList, getRapTapLevels(me, whichuser)
append GameLevelList, getBalloonLevels(me, whichuser)
append GameLevelList, getCoalCarLevels(me, whichuser)
closeRecords gRecordKeeper
return GameLevelList
end

on getBalloonLevels me, whichUser
  set BalloonLevels = [0,0,0,0]
  set PrefsMember = whichUser&"7"
  put value (line 11 of field PrefsMember of castLib CastLibname) into levelsList
  -- get completed levels list for Karloons - game 5
  sort LevelsList

  if getOne(levelsList,6) then
    setAT BalloonLevels, 1, 2
  else if getOne(levelsList,3) then
    setAT BalloonLevels, 1, 1
  end if

  if getOne(levelsList,18) then
    setAT BalloonLevels, 2, 2
  else if getOne(levelsList,12) then
    setAT BalloonLevels, 2, 1
  end if

  if getOne(levelsList,30) then
    setAT BalloonLevels, 3, 2
  else if getOne(levelsList,24) then
    setAT BalloonLevels, 3, 1
  end if

  if getOne(levelsList,38) then
    setAT BalloonLevels, 4, 2
  else if getOne(levelsList,34) then
    setAT BalloonLevels, 4, 1
  end if

  return balloonLevels
end

on getEggBasketLevels me, whichuser
  set EggBasketLevels = [0,0,0,0,0,0,0]
  set PrefsMember = whichUser&"7"
  put value (line 8 of field PrefsMember of castLib CastLibname) into levelsList
  -- get completed levels list for Karloons - game 2
  sort LevelsList

  if getOne(levelsList,3) then
    setAT EggBasketLevels, 1, 1
  end if
  if getOne(levelsList,11) then
    setAT EggBasketLevels, 2, 1
  end if
  if getOne(levelsList,20) then
    setAT EggBasketLevels, 3, 1
  end if

```

```

if getOne(levelsList,30) then
  setAT EggBasketLevels, 4, 1
end if

if getOne(levelsList,58) then
  setAT EggBasketLevels, 5, 4
else if getOne(levelsList,51) then
  setAT EggBasketLevels, 5, 3
else if getOne(levelsList,44) then
  setAT EggBasketLevels, 5, 2
else if getOne(levelsList,37) then
  setAT EggBasketLevels, 5, 1
end if

if getOne(levelsList,86) then
  setAT EggBasketLevels, 6, 4
else if getOne(levelsList,79) then
  setAT EggBasketLevels, 6, 3
else if getOne(levelsList,72) then
  setAT EggBasketLevels, 6, 2
else if getOne(levelsList,65) then
  setAT EggBasketLevels, 6, 1
end if

if getOne(levelsList,114) then
  setAT EggBasketLevels, 6, 4
else if getOne(levelsList,107) then
  setAT EggBasketLevels, 6, 3
else if getOne(levelsList,100) then
  setAT EggBasketLevels, 6, 2
else if getOne(levelsList,93) then
  setAT EggBasketLevels, 6, 1
end if

return EggBasketLevels

end

on getCoalCarLevels me, whichuser
  set CoalCarLevels = [0,0,0,0,0,0,0]
  set PrefsMember = whichUser&"7"
  put value (line 12 of field PrefsMember of castLib CastLibname) into levelsList
  -- get completed levels list for coal car game 6
  sort LevelsList

  if getOne(levelsList,10) then
    setAT CoalCarLevels, 1, 1
  end if

  if getOne(levelsList,20) then
    setAT CoalCarLevels, 2, 1
  end if

  if getOne(levelsList,34) then
    setAT CoalCarLevels, 3, 1
  end if

  if getOne(levelsList,40) then
    setAT CoalCarLevels, 4, 1
  end if

```

```

if getOne(levelsList,52) then
    setAt CoalCarLevels, 5, 1
end if

if getOne(levelsList,56) then
    setAt CoalCarLevels, 6, 1
end if

if getOne(levelsList,74) then
    setAt CoalCarLevels, 7, 4
else if getOne(levelsList,72) then
    setAt CoalCarLevels, 7, 3
else if getOne(levelsList,66) then
    setAt CoalCarLevels, 7, 2
else if getOne(levelsList,63) then
    setAt CoalCarLevels, 7, 1
end if

return CoalCarLevels
end

on getCaterPillarLevels me, whichuser
    set CaterPillarLevels = {0,0,0,0,0,0}
    set PrefsMember = whichUser&"7"
    set LevelsList = []
    put value (line 7 of field PrefsMember of castLib CastLibname) into levelsList
    -- get completed levels list for game 1
    sort LevelsList

    if getOne(LevelsList, 12) then
        setAt CaterPillarLevels, 1, 3
    else if getOne(LevelsList, 8) then
        setAt CaterPillarLevels, 1, 2
    else if getOne(LevelsList, 4) then
        setAt CaterPillarLevels, 1, 1
    end if

    if getOne(LevelsList, 24) then
        setAt CaterPillarLevels, 2, 3
    else if getOne(LevelsList, 20) then
        setAt CaterPillarLevels, 2, 2
    else if getOne(LevelsList, 16) then
        setAt CaterPillarLevels, 2, 1
    end if

    if getOne(LevelsList, 28) then
        setAt CaterPillarLevels, 3, 1
    end if

    if getOne(LevelsList, 32) then
        setAt CaterPillarLevels, 4, 1
    end if

    if getOne(LevelsList, 44) then
        setAt CaterPillarLevels, 5, 3
    else if getOne(LevelsList, 40) then
        setAt CaterPillarLevels, 5, 2
    else if getOne(LevelsList, 36) then
        setAt CaterPillarLevels, 5, 1
    end if

```

```

if getOne(LevelsList, 56) then
    setAt CaterPillarLevels, 6, 3
else if getOne(LevelsList, 52) then
    setAt CaterPillarLevels, 6, 2
else if getOne(LevelsList, 48) then
    setAt CaterPillarLevels, 6, 1
end if

return CaterPillarLevels
end

on getRapTapLevels me, whichuser
    set RapTapLevels = {0,0,0,0}
    set PrefsMember = whichUser&"7"
    put value (line 10 of field PrefsMember of castLib CastLibname) into levelsList
    -- get completed levels list for game 4
    sort LevelsList

    if getOne(LevelsList,6) then
        setAt RapTapLevels,1,6
    else if getOne(LevelsList,5) then
        setAt RapTapLevels,1,5
    else if getOne(LevelsList,4) then
        setAt RapTapLevels,1,4
    else if getOne(LevelsList,3) then
        setAt RapTapLevels,1,3
    else if getOne(LevelsList,2) then
        setAt RapTapLevels,1,2
    else if getOne(LevelsList,1) then
        setAt RapTapLevels,1,1
    end if

    if getOne(LevelsList,12) then
        setAt RapTapLevels,2,6
    else if getOne(LevelsList,11) then
        setAt RapTapLevels,2,5
    else if getOne(LevelsList,10) then
        setAt RapTapLevels,2,4
    else if getOne(LevelsList,9) then
        setAt RapTapLevels,2,3
    else if getOne(LevelsList,8) then
        setAt RapTapLevels,2,2
    else if getOne(LevelsList,7) then
        setAt RapTapLevels,2,1
    end if

    if getOne(LevelsList,14) then
        setAt RapTapLevels,3,2
    else if getOne(LevelsList,13) then
        setAt RapTapLevels,3,1
    end if

    if getOne(LevelsList,16) then
        setAt RapTapLevels,4,2
    else if getOne(LevelsList,15) then
        setAt RapTapLevels,4,1
    end if

```

```

return RapTapLevels
end

on getRhymeTimeLevels me, whichuser
    set RhymeTimeLevels = [0,0]
    set PrefsMember = whichUser&"7"
    put value (line 9 of field PrefsMember of castLib CastLibname) into levelsList
    -- get completed levels list for game 3
    sort LevelsList

    if getOne(LevelsList,5) then
        setAt RhymeTimeLevels,1,5
    else if getOne(LevelsList,4) then
        setAt RhymeTimeLevels,1,4
    else if getOne(LevelsList,3) then
        setAt RhymeTimeLevels,1,3
    else if getOne(LevelsList,2) then
        setAt RhymeTimeLevels,1,2
    else if getOne(LevelsList,1) then
        setAt RhymeTimeLevels,1,1
    end if

    if getOne(LevelsList,11) then
        setAt RhymeTimeLevels,2,6
    else if getOne(LevelsList,10) then
        setAt RhymeTimeLevels,2,5
    else if getOne(LevelsList,9) then
        setAt RhymeTimeLevels,2,4
    else if getOne(LevelsList,8) then
        setAt RhymeTimeLevels,2,3
    else if getOne(LevelsList,7) then
        setAt RhymeTimeLevels,2,2
    else if getOne(LevelsList,6) then
        setAt RhymeTimeLevels,2,1
    end if
    return RhymeTimeLevels
end

on new me
    global gCastLibName
    -- Pub.
    set CastLibName = gCastLibName
    set myHandlers = 0
    set myHandlers = GetMyHandlers(me)

    return me
end

on xx-----Private Handlers-----
    -- i'm a separator
end

on GetMyHandlers me
    -- Priv
    -- returns list of handlers to property variable

```



```

put value(word 2 of string(me)) into whichCast
put the scripttext of member whichCast into text

put the number of lines in text into scriptLines
Put whichCast && "handlers"& return into Handlers

```

```

repeat with x = 1 to scriptLines
    if word 1 of line x of text = "on" then
        put line x of text into handlerName
        delete word 1 of handlerName
        put handlerName & return after handlers
    end if
end repeat
return handlers
end

```

```

on xxx-----Testing Handlers-----
    -- i'm a separator
    nothing
end

```

```

on showHandlers me
    -- Testing
    -- puts list of handlers in message window
    put myHandlers
end

```

```

on showProps me
    -- testing
    -- puts list of properties and their current values in message window
    set PropNum = count(me)
    repeat with x = 1 to PropNum
        set prop = 0
        set thisProp = getpropat(me, x)
        if thisProp = #myHandlers then next repeat
        put (string (getpropat(me, x))) &&"="&& getaProp(me, thisProp) into prop
        put prop
    end repeat
end

```

Movie Script4:--DataViewPrefsManager

Property ancestor, pDateFormat, pCastLibName

on new me

global gRecordKeeper

set ancestor = gRecordKeeper

set pCastLibName = the castLibName of ancestor

return me

end

on getDateFormat me

-- returns 0,1,or 2 which is stored in line 1 of

-- dataView Prefs cast. If cast not there it makes on

-- and returns 0 to start as a default.

-- 0 means MM/DD/YY

-- 1 means DD/MM/YY

-- 2 means YY/MM/DD

openrecords me

if the number of member "dataViewPrefs" of castLib pCastLibName = -1 then

-- no prefs so make new one

new #field, member 5 of castLib pCastLibName

put 0 into line 1 of field 5 of castLib pCastLibName

set the name of member 5 of castLib pCastLibName = "DataViewPrefs"

set pDateFormat = 0

saveRecords me

CloseRecords me

return 0

else

set pDateFormat = line 1 of field "DataViewPrefs" of castLib pCastLibName

closeRecords me

return pDateFormat

end if

end

Score Script5

on exitFrame

go to the frame

end

Parent Script6:recordDisplay

-- this object is designed to take the data from the "records.cst" cast and format and
-- display that data on the screen

property CastLibName, myHandlers, ancestor

on x-----Public Handlers -----

-- I'm a separator
end

on new me

global gRecordKeeper

if objectP(gRecordKeeper) then

set ancestor to gRecordKeeper

else

alert "Be sure to create recordKeeper object before creating recordDisplay object"

abort

end if

set castLibName = "records.cst"

set myHandlers = 0

set myHandlers = GetMyHandlers(me)

return me

end

on xx-----Private Handlers-----

-- i'm a separator

end

on GetMyHandlers me

-- Priv.

-- returns list of handlers to property variable

put value(word 2 of string(me)) into whichCast

put the scripttext of member whichCast into text

put the number of lines in text into scriptLines

Put whichCast && "handlers" & return into Handlers

repeat with x = 1 to scriptLines

if word 1 of line x of text = "on" then

put line x of text into handlerName

delete word 1 of handlerName

put handlerName & return after handlers

end if

end repeat

return handlers

end

on xxx-----Testing Handlers-----

-- i'm a separator

nothing

end

on showHandlers me

-- Testing

-- puts list of handlers in message window

put myHandlers

end

```

on showProps me
  -- testing
  -- puts list of properties and their current values in message window
  set PropNum = count(me)
  repeat with x = 1 to PropNum
    set prop = 0
    set thisProp = getpropat(me, x)
    if thisProp = #myHandlers then next repeat
    put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop
    put prop
  end repeat
end

```

Score Script7

```

on exitFrame
  pause
end

```

Movie Script8

```

on PurgeRecords
  -- testing
  -- use to clean out all records! only called during authoring
  -- use carefully, there is no undo!!
  global gCastLibName
  if not (the optionDown) then
    alert "Are you sure you want to empty all records? This cannot be
undone!" & return & return & "To proceed, issue this command while holding down the option
key."
    exit
  end if
  if the optionDown then
    openrecords me
    repeat with x = 1 to 2
      put "" into field x of castLib gCastLibName
    end repeat
    repeat with x = 11 to the number of members of castLib gCastLibName
      erase member x of castLib gCastLibName
    end repeat
    saveRecords me
    closeRecords me
    put "Records Purged"
  end if
end

```

Score Script9

```

on exitFrame
-- global goodCD,jimx,grecordkeeper,trycdagain
-- if the machineType <> 256 then
--   set the colorDepth = 8
--   set the searchCurrentFolder to true
--
--   put getNthFileNameInFolder("CCI:Datatest:",1) into x
--   if x = "02.Cst" or x = "02.Cxt" then
--     put getNameNumber(grecordkeeper) into serial
--     if serial <> "" then
--       go to movie "CCI:Datatest:DATATEST"
--       exit
--     else
--       go to movie "CCI:DataTest:License"
--       exit
--     end if
--
--   else
--     alert "Please Insert the Earobics PRO PLUS CD"
--     if trycdagain < 2 then
--       set trycdagain = trycdagain + 1
--       go to the frame
--     else
--       halt
--     end if
--
--   end if
--
-- else
--   set cdList =
list("c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w",
"x","y","z")
--   repeat with DriveNum in cdlist
--     put DriveNum & ":\DataTest\" into TryThis
--     put getNthFileNameInFolder(TryThis,1) into x
--     if x = "02.Cst" or x = "02.Cxt" then
--       set goodCD = True
--       put getNameNumber(grecordkeeper) into serial
--       if serial <> "" then
--         go to movie DriveNum & ":\DataTest\DATATEST"
--         exit
--       else
--         go to movie DriveNum & ":\DataTest\License"
--         exit
--       end if
--
--     end if
--   end repeat
--
-- if GoodCD = False then
--   alert "Please Insert the Earobics PRO PLUS CD"
--   if trycdagain < 2 then
--     set trycdagain = trycdagain + 1
--     go to the frame
--   else
--     halt
--   end if
-- end if

```

```
-- end if
--
-- go to movie "earobicsv2:DataTest:DataTest"
go to movie "cciOG:DataTest:License"
-- go to the frame
end
```


Movie Script1:StartMovieScript

-- version as of 4/3/97

on startMovie

global gMenuMaker, gRecordDisplay, gGameList, gRTFieldList, gSessionNum,
gBalloonFields, gTheFont, gEgg2ob, gBllnOb, gDataViewPrefsMan
global gCatConFields, gNextButton, gPrevButton, gMoreButtonLoc
global gMoreButton, gDataPrintButton, jimWhichGame, jimWhichUser, gGamesViewedList

set the itemDelimiter = ","

puppetsound 1,0

puppetsound 2,0

cursor 4

initDataFieldLists

set gSessionNum = 1

set gMenuMaker = 0

set gMenuMaker = new(script "MenuMaker")

set gRecordDisplay = 0

set gRecordDisplay = new(script "NewrecordDisplay")

set gDataViewPrefsMan = 0

set gDataViewPrefsMan = new(script "dataViewPrefsManager")

if the machineType = 256 then

set gTheFont = "arial"

else

set gTheFont = "helvetica"

end if

InitMenuFields

set gGameList = []

repeat with x = 1 to the number of lines in field "gameList"

set game = line x of field "gameList"

if game = "" then exit repeat

append gGameList, game

end repeat

set gNextButton = 25 -- sprite of "LeftArrow"

set gPrevButton = 27 -- sprite of "RightArrow"

set gMoreButton = 28 -- "moreData" button sprite

set gDataPrintButton = 29 -- sprite of dataPrint button

set gMoreButtonLoc = point(182,467) -- location of "moreData" button when needed on stage

setFieldRects

set gGamesViewedList = []

set jimWhichGame = ""

set jimWhichUser = ""

set gEgg2ob = new (script "Egg2FieldMemberNums")

set gBllnOb = new (script "BalloonFieldMemberNums")

register(xtra "printomatic", "POMX153-501-02580")

put empty before line 1 of field "IntroText"

cursor -1

end

on initDataFieldLists

global gRTFieldList, gBalloonfields, gCatConFields, gEggTask1Fields, gEggTask2Fields

global gRapTapFields, gCoallFields, gCoal2Fields

set gRTFieldList = ["DataDate", "RT Task(1)", "RT InSetOf", "RT BgNoise1", "RT Score1", "RT


```

Task(2)", "RT responseChoice", "RT BgNoise2", "RT Score2"]

set gBalloonfields = ["DataDate", "BalloonTask", "BalloonNumber", "BalloonStimType",
"BalloonVisDisplay", "BalloonNoise", "BalloonScore"]

set gCatConFields = ["DataDate", "CatConTask", "CatConUnits", "CatConInterval",
"CatConTarget", "CatConFails", "CatConScore"]

set gEggTask1Fields =
["DataDate", "EggTask1", "EggVowels", "Egg1DiffScore", "Egg1SameScore"]

set gEggTask2Fields = ["DataDate", "EggTask2", "EggDuration", "EggAmplification",
"EggSteps", "Egg2DiffScore", "Egg2SameScore"]

-- set gRapTapFields = ["DataDate", "RapTapTask1", "RapTap1Units", "RapTap1Stimulus",
"RapTap1Interval", "RapTap1FeedBack", "RapTap1Score", "RapTapTask2",
"RapTap2Units", "RapTap2Stimulus", "RapTap2FeedBack", "RapTap2Score"]

set gRapTapFields = ["DataDate": "Date:", "RapTapTask1": "Task (1):",
"RapTap1Units": "Units:", "RapTap1Stimulus": "Stimuli:", "RapTap1Interval": "Interval:",
"RapTap1FeedBack": "Feedback:", "RapTap1Score": "Cuml. Score:", "RapTapTask2": "Task (2):",
"RapTap2Units": "Units:", "RapTap2Stimulus": "Stimulus:", "RapTap2FeedBack": "Auditory
Feedback:", "RapTap2Score": "Cuml. Score:"]

set gCoal1Fields = ["DataDate", "CoalTask1", "Coal1Context", "Coal1Score"]
set gCoal2Fields = ["DataDate", "CoalTask2", "Coal2TargPho", "Coal2Score"]

end

on stopMovie
global gVoid, gMenuMaker, gRecordDisplay, gGameList, gRTFieldList, gSessionNum,
gTheFont
global gWhichUser, gWhichGame, gCatConFields, gBalloonFields, gNextButton, gPrevButton,
gNextStart, gMoreButtonLoc, gRapTapKeyWords
global gMoreButton, gSessionList, gEggTask1Fields, gEggTask2Fields, gCoal1Fields,
gCoal2Fields
global gRapTapFields, gGamesViewedList, gOverRunSession, gNumPages, gSpritesOnList
global gDataViewPrefsMan, gEgg2ob, gBlinOb, gPrintMan, gDataPrintButton

set gGamesViewedList = gVoid
set gSpritesOnList = {}
if the runmode <> "author" then
set gDataViewPrefsMan = gVoid
set gMenuMaker = gVoid
set gRecordDisplay = gVoid
set gGameList = gVoid
set gRTFieldList = gVoid
set gSessionNum = gVoid
set gWhichUser = gVoid
set gWhichGame = gVoid
set gBalloonFields = gVoid
set gCatConFields = gVoid
set gNextButton = gVoid
set gPrevButton = gVoid
set gMoreButtonLoc = gVoid
set gDataPrintButton = gVoid
set gnextStart = gVoid
set gMoreButton = gVoid
set gSessionList = gVoid

```



```

set gEggTask1Fields = gVoid
set gEggTask2Fields = gVoid
set gRapTapFields = gVoid
set gCoallFields = gVoid
set gCoal2Fields = gVoid
set gGamesViewedList = gVoid
set gTheFont = gVoid
set gEgg2ob = gVoid
set gBllnOb = gVoid
set gOverRunSession = gVoid
set gNumPages = gVoid
set gPageNum = gVoid
set gRapTapKeywords = gVoid
set gPrintMan = gVoid
end if

end

on goJim
  global gWhichGame, gsessionNum
  cursor 4 -- for quicker cursor change during data reads
  puppetSprite 39, false
  puppetSprite 40, false
  puppetSprite 41, false
  initDataFrame gWhichGame
  set whichFrame = gwhichGame&&"Data"
  set gSessionNum = 1
  spriteListOff
  go to frame whichFrame
  -- initDataFrame gWhichGame
end

```

Parent Script2:menuMaker

property BlackDot

on new me

set BlackDot = the memberNum of member "blackDot"

return me

end

on showmenu me, menufield, where

set the member of sprite 40 to menufield

put the height of member menufield into fH

put the textheight of member menufield into tH

set the height of sprite 39 to the height of member menufield - tH

set the width of sprite 39 to the width of member menufield

set the loc of sprite 40 to where

set the loc of sprite 39 to where

updatestage

put the top of sprite 40 into V

set the width of sprite 41 to (the width of member menufield)

set the height of sprite 41 to tH

put (the locH of sprite 40) into hLoc

repeat while stillDown()

if mouseCast() = menufield or mousecast() = BlackDot then

put (mouseV() - V) - ((mouseV() - V) mod tH) into vLoc

set the loc of sprite 41 to point(hLoc, constrainV (39, V + vLoc))

updatestage

else

set the locH of sprite 41 to 10000

updatestage

end if

end repeat

set the locH of sprite 41 to 10000

updatestage

if mouseCast() <> menufield then -- didn't select anything

set the locH of sprite 40 to 10000

set the locH of sprite 39 to 10000

updatestage

return #Nothing

end if

put the mouseLine into theLine

wait 2

set the loc of sprite 41 to point(hLoc, constrainV (39, V + vLoc))

updatestage

wait 2

repeat with x = 1 to 2

set the locH of sprite 41 to 10000

updatestage

wait 2

set the loc of sprite 41 to point(hLoc, constrainV (39, V + vLoc))

updatestage

wait 2

end repeat

set the locH of sprite 41 to 10000

wait 7

updatestage

```
-- put the mouseLine into theLine
set the locH of sprite 40 to 10000
set the locH of sprite 39 to 10000
updatestage
return theLine
end
```

```
-- sample castScript for menuMember
```

```
--on mouseDown
-- global MenuMaker
-- set the ink of sprite 4 to 2
-- updatestage
-- set theLoc to point(the left of sprite 4, the bottom of sprite 4)
-- showMenu menuMaker, 4, theLoc
-- set the ink of sprite 4 to 0
-- updatestage
--end
```

Parent Script3:NewrecordDisplay

```
--2/3/98
-- changed way lists are broken up for two levels

-- 4/7/97
-- Changed reporting so list of sessions is generated last to first not first to last
-- this object is designed to take the data from the "records.cst" cast and format and
-- display that data on the screen

property CastLibName, myHandlers, ancestor, lastRecord, lastDataList

on x-----Public Handlers -----
  -- I'm a separator
end

on new me
  global gRecordKeeper
  if objectP(gRecordKeeper) then
    set ancestor to gRecordKeeper
  else
    alert "Be sure to create recordKeeper object before creating recordDisplay object"
    abort
  end if
  set castLibName = "records.cst"
  set myHandlers = 0
  set myHandlers = GetMyHandlers(me)
  return me
end

on GetData me, whichUser, WhichGame, forPrintOut-- function
  -- Pub.
  -- This handler when given a userName (from the list in
  -- member 1 of castLib "records.cst" and either a game name or
  -- integer game number from 1 to 7 opens the record for that
  -- user and that game and reads that data into a variable.
  -- it then loops through that text, line by line, and creates
  -- a property list with property "date" (from the session dates
  -- stored in record) associated with another propList as it's
  -- value. This second list contains the levels for that date's session
  -- and each level's corresponding score converted to a string
  -- percentage. That whole list is then returned to the calling handler
  global gGameList
  if stringP(WhichGame) then
    set gameNum = getone(gGameList, whichGame)
    if gameNum = 0 then
      alert "This game is not in the Game List. Check your spelling and your Lingo"
      abort
    end if
  end if
  if integerP(whichGame) then
    if whichGame < 1 or whichGame > 7 then
      alert "Expecting an integer from 1 to 7. Check your Lingo."
      abort
    end if
    set gameNum = whichGame
  end if
```

```

-- after error checking we open the records and read it into a variable
set thisRecord = whichUser & gameNum
if thisRecord = lastRecord and voidP(forPrintOut) then--2/3/98
  -- if forPrintOut we need to re-order some of the data!
  return lastDataList
else
  openRecords me
  set record = field thisrecord of castLib CastLibName
  closeRecords me

  set DataList = [:] -- init lists for line by line readthrough
  set Dates = []
  set LevelsAndScores =[:]

  set numLines = the number of lines of record
  -- repeat with x = 3 to numlines
  repeat with x = numlines down to 3 -- changed 4/7/97
    set thisLine = line x of record
    if thisLine = "" then next repeat -- check for empty lines (spacers)
    put item 1 of thisLine into Date -- get stored date
    if date <> getLast(dates) then
      -- see if date is already in list "Dates"
      -- if so skip down to collecting levels and scores
      -- if not then check if this is first date encountered
      if count(dates) > 0 then
        -- if list already has items then we are hitting our second date
        -- so store the list of levels and scores (after sorting first)
        -- we've been making and pair it with its date.
        -- if not skip this stuff and just store the new date
        -- and init the list for LevelsandScores
        sort levelsAndScores
        addProp dataList, getLast(dates), levelsAndScores
      end if
      -- store the date and init list for levels and scores
      append dates, date
      set LevelsAndScores =[:]
    end if

    -- here we loop through the lines, figuring the percents and
    -- adding to the levelsAndScores list [level:percent]

    set Level = Value(item 2 of thisLine)

    -- 2/3/98
    -- if count(levelsAndScores) and GameNum = 6 then
    --   -- Here we need to check if user is in game 6 and then check
    --   -- if user played both tasks in one session
    --   -- if so we break that session into two sessions
    --   set tempCount = count(levelsAndScores)
    --   if getPropAt(levelsAndScores,tempcount) >=57 and level <=56 then
--4/7/97
    --   -- we have a session with plays in both tasks of the game so
    --   -- split off lower list and start new list for upper part
    --   sort levelsAndScores
    --   addProp dataList, getLast(dates), levelsAndScores
    --   set LevelsAndScores =[:]
    --   end if
    -- end if

```



```

    set numRight = value(item 4 of thisLine)
    set numPlays = value(item 3 of thisLine)
    set score = Float(numRight)/Float(numPlays)
    set score = integer(score*100)
    set score = string(score)&"%"
    addProp LevelsAndScores, level, score
end repeat
-- last time we need to add the last list here
set LevelsAndScores
addProp dataList, date, LevelsAndScores

if gameNum = 6 then -- new way to break list appart!! (2.3/98)
    set dataList = convertList(dataList, 57, forPrintOut)
end if
if voidP(forPrintOut) then -- new 2/3/98
    -- don't save data to property if called
    -- from printout
    set lastRecord = thisRecord
    set lastDataList = dataList
end if
return dataList
end if
end

on getEggBasketData me, whichUser, forPrintOut
    -- Unlike the other games egg basket stores two kinds of data
    -- for each round, the number of plays and number of right responses
    -- when then two test sounds are the same and the numbers when the
    -- two test sounds are different. Items 3 and 4 on each line store the
    -- "different" data and items 5 and 6 store the "same" data.
    set thisRecord = whichUser & 2
    if thisRecord = lastRecord and voidP(forPrintOut) then -- 2/3/98
        -- if forPrintOut we need to re-order some of the data!
        return lastDataList
    else
        openRecords me
        set record = field thisrecord of castLib CastLibName
        closeRecords me

        set DataList = [:] -- init lists for line by line readthrough
        set Dates = []
        set LevelsAndScores = [:]

        set numLines = the number of lines of record
        -- repeat with x = 3 to numlines
        repeat with x = numlines down to 3 -- changed 4/7/97
            set thisLine = line x of record
            if thisLine = "" then next repeat -- check for empty lines (spacers)
            put item 1 of thisLine into Date -- get stored date
            if date <> getlast(dates) then
                -- see if date is already in list "Dates"
                -- if so skip down to collecting levels and scores
                -- if not then check if this is first date encountered
                if count(dates) > 0 then
                    -- if list already has items then we are hitting our second date
                    -- so store the list of levels and scores (after sorting first)
                    -- we've been making and pair it with its date.
                    -- if not skip this stuff and just store the new date
                    -- and init the list for LevelsAndScores

```

```

        sort levelsAndScores
        addProp dataList, getLast(dates), levelsAndScores
    end if
    -- store the date and init list for levels and scores
    append dates, date
    set LevelsAndScores = [:]
end if

-- here we loop through the lines, figuring the percents and
-- adding to the levelsAndScores list [level:[diffPercent,samePercent]]

set Level = Value(item 2 of thisLine)
-- 2/3/98
--     if count(levelsAndScores) then
--         set tempCount = count(levelsAndScores)
--
--         if getPropAt(levelsAndScores,tempcount) >=31 and level <=30 then
-- 4/7/97
--             -- we have a session with plays in both tasks of the game so
--             -- split off lower list and start new list for upper part
--             sort levelsAndScores
--             addProp dataList, getLast(dates), levelsAndScores
--             set LevelsAndScores = [:]
--             end if
--         end if
set NumRightDiff = value(item 4 of thisLine)
set NumPlaysDiff = value(item 3 of thisLine)
if NumPlaysDiff = 0 then
    set DiffScore = "---"
else
    set DiffScore = Float(NumRightDiff)/Float(NumPlaysDiff)
    set DiffScore = integer(DiffScore*100)
    set DiffScore = string(DiffScore)&"%"
end if

set NumRightSame = value(item 6 of thisLine)
set NumPlaysSame = value(item 5 of thisLine)
if NumPlaysSame = 0 then
    set SameScore = "---"
else
    set SameScore = Float(NumRightSame)/Float(NumPlaysSame)
    set SameScore = integer(SameScore*100)
    set SameScore = string(SameScore)&"%"
end if

set scores = []
append scores DiffScore
append scores SameScore

addProp LevelsandScores, level, scores

end repeat
-- last time we need to add the last list here
sort levelsAndScores
addProp dataList, date, levelsAndScores
set dataList = convertList(dataList,31,forPrintOut) -- new 2/3/98
if voidP(forPrintOut) then -- new 2/3/98
    -- don't save data to property if called
    -- from printout
    set lastRecord = thisRecord
    set LastDataList = dataList

```



```

    end if
    return dataList
end if
end

```

```

end

```

```

on GetHighLevel me, whichUser, whichGame -- (function)
-- Pub.
global gGameList
if stringP(WhichGame) then
set gameNum = getone(gGameList, whichGame)
if gameNum = 0 then
alert "This game is not in the Game List. Check your spelling and your Lingo"
abort
end if
end if
if integerP(whichGame) then
if whichGame < 1 or whichGame > 7 then
alert "Expecting an integer from 1 to 7. Check your Lingo."
abort
end if
set gameNum = whichGame
end if
set thisRecord = whichUser & whichGame
openrecords me
put item 3 of line 1 of field thisRecord of castLib castLibName into level
if level = "" then set level = 1
put "Highest level of " & whichUser & " on game " & whichGame & " is " & level
return level
closerecords me
end

```

```

on xx-----Private Handlers-----
-- i'm a separator
end

```

```

on GetMyHandlers me
-- Priv.
-- returns list of handlers to property variable
put value(word 2 of string(me)) into whichCast
put the scripttext of member whichCast into text

put the number of lines in text into scriptLines
Put whichCast & "handlers" & return into Handlers

repeat with x = 1 to scriptLines
if word 1 of line x of text = "on" then
put line x of text into handlerName
delete word 1 of handlerName
put handlerName & return after handlers
end if
end repeat
return handlers
end

```

on xxx-----Testing Handlers-----

-- i'm a separator

nothing

end

on showHandlers me

-- Testing

-- puts list of handlers in message window

put myHandlers

end

on showProps me

-- testing

-- puts list of properties and their current values in message window

set PropNum = count(me)

repeat with x = 1 to PropNum

set prop = 0

set thisProp = getpropat(me, x)

if thisProp = #myHandlers then next repeat

put (string (getpropat(me, x))) &&"=" && getaProp(me, thisProp) into prop

put prop

end repeat

end

Score Script4

Movie Script5

on getRapTapRects

global gRapTapFields

put " " into field "tempRects"

set y = count(gRapTapFields)

repeat with x = 1 to y

set thisfield = getpropat(gRapTapFields, x)

put "set the rect of member"&&Quote&thisfield&Quote&&"=" && -

the rect of member thisfield & return after field "tempRects"

end repeat

end

Movie Script10

```
on InitMenuFields
    global gRecordDisplay, gTheFont
    set MenuFieldList = ["UserNameList", "GameList"]
    put " " into field "userNameList"
    put getUserNames (gRecordDisplay) into field "userNameList"
    -- put field "userNameListx" into field "userNameList" -- testing
    repeat with thisField in menuFieldList
        set the font of field thisField = gTheFont
        set the textStyle of field thisField = "Bold"
        set the textHeight of field thisField = 14
        set the border of member thisField = 1
        set the boxDropShadow of member thisField = 1
    end repeat
    put the number of lines of field "userNameList" into numLines
    repeat with x = 1 to numLines
        -- put in a space at start of the name to make more readable
        -- remember to take it back out !!
        put " " & line x of field "userNameList" into line x of field "userNameList"
        if line x of field "userNameList" = " " then delete line x of field "userNameList"
    end repeat
end
```

Score Script11

```
on exitFrame
    global jimWhichGame, jimWhichUser
    puppetSprite 39, true
    puppetSprite 40, true
    puppetSprite 41, true
    set jimWhichGame = ""
    set jimWhichUser = ""
    cursor -1
end
```

Score Script12

```
on exitFrame
  go to the frame
  cursor - 1
end
```

Script of Cast Member13

```
on mouseUp
  global gWhichGame, gsessionNum
  puppetSprite 39, false
  puppetSprite 40, false
  puppetSprite 41, false
  initDataFrame gWhichGame
  set whichFrame = gWhichGame&&"Data"
  set gSessionNum = 1
  go to frame whichFrame
end
```

Score Script14

```
on exitFrame
  cursor - 1
  checkDataPrintButton
  go to the frame
end
```

Score Script17

```
on enterFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  spritesonList[10,11,12,13,15,16,17,18]
  puppetButtonSprites true
  initRtFields

end
```

Movie Script27

```
on initDataFrame whichGame
  global gWhichGame, gWhichUser
  put gWhichGame & "Data Report Table" into field 1 "DataTitle"
  put " " into field "Specifics"
  set the fontstyle of field "Specifics" = "Plain"
  put gWhichUser into line 1 of field "Specifics"
  put gWhichGame into line 2 of field "Specifics"
  cursor 4
  case (gWhichGame) of
    "Rhyme Time": put "Rhyming, Figure-Ground Discrimination, Attention & Memory" into
line 3 of field "Specifics"

    initRTFields
    "Caterpillar Connection": put "Auditory Synthesis, Attention & Memory; Phonological
Awareness" into line 3 of field "Specifics"
    initCatConFields
    "Karloon's Balloons": put "Auditory Memory, Figure-Ground Discrimination,
Identification, &RETURN&Discrimination & Attention." into line 3 of field "Specifics"
    initBalloonfields
    "Basket Full of Eggs": put "Auditory Discrimination, Attention & Memory" into line 3
of field "Specifics"

    "Rap-A-Tap-Tap": put "Auditory Segmentation, Attention & Memory; Phonological
Awareness" into line 3 of field "Specifics"
    initRapTapFields
    "C.C. Coal Car": put "Phonological Awareness, Sound-Symbol Correspondence, Auditory
Identification, Discrimination, Attention & Memory " into line 3 of field "Specifics"
  end case

-- cursor -1
end
```

Movie Script42

```
on initRTFields -- empty rhyme time's data fields of all but first Line
  global gRTFieldList, gGamesViewedList, gTheFont
  if not getOne(gGamesViewedList, #RhymeTime) then -- check if we've been here yet this
session
    append gGamesViewedList, #RhymeTime
    repeat with thisfield in gRTFieldList
      set x = the number of lines of field thisField
      set the font of field thisfield = gTheFont
      set the fontsize of field thisfield = 12

      case (thisField) of
        "DataDate":Put "Date:" into text
        "RT Task(1)":Put "Task (1):" into text
        "RT Task(2)":Put "Task (2):" into text
        "RT InSetOf":Put "In set of:" into text
        "RT BgNoise1":Put "Background Noise:" into text
        "RT BgNoise2":Put "Background Noise:" into text
        "RT responseChoice":Put "Response Choices:" into text
        "RT Score1":Put "Cuml. Score:" into text
        "RT Score2":Put "Cuml. Score:" into text
      end case

      put text & return into field thisField
      set the fontStyle of Line 1 of field thisfield = "underline"
      delete Line 2 to x of field thisField
      put " " into line 2 of field thisfield
      setPlainStyle(2, thisField)
    end repeat
  else
    repeat with thisfield in gRTFieldList
      -- if Line 2 of field thisField = " " then exit repeat
      set x = the number of lines of field thisField
      delete Line 2 to x of field thisField
      put " " into line 2 of field thisfield
      setPlainStyle(2, thisField)
    end repeat
  end if
  if label(0) = label("Rhyme Time Data") then
    set LowerFieldSprites = {15,16,17,18}
    repeat with x in lowerFieldSprites
      set the loc of sprite x to point(1000,1000)
    end repeat
    updatestage
  end if
end
```



```

on spritesOnList Spritelist
  -- Takes a LIST of non-consecutive (or consecutive) channels
  -- and puppets them. Must be passed as a list []
  -- turns global list gSpritesOnList off first and
  -- then turns on spritelist and makes SpriteList
  -- into gSpritesOnList
  global gSpritesOnList
  if voidP(gSpritesOnList) then set gSpritesOnList = []

  if count(gSpritesOnList) > 0 then
    repeat with thisSprite in gSpritesOnList
      puppetsprite (thisSprite, false)
    end repeat
    repeat with thisSprite in spritelist
      puppetsprite thisSprite, true
    end repeat
    set gSpritesOnList = spritelist
  else
    repeat with thisSprite in spritelist
      puppetsprite thisSprite, true
    end repeat
    set gSpritesOnList = spritelist
  end if
end

on spriteListOff
  -- turns off all sprites on Current gSpritesonList
  -- and re-initializes that global
  global gSpritesOnList
  if voidP(gSpritesOnList) then
    set gSpritesOnList = []
    exit
  end if
  repeat with thisSprite in gSpritesonList
    puppetsprite thisSprite, false
  end repeat
  set gSpritesOnList = []
end

on spriteson FirstSprite, LastSprite
  -- turns on sprites in consecutive channels from
  -- FirstSprite to LastSprite
  global gSpritesOnList
  if count(gSpritesOnList) > 0 then
    repeat with thisSprite in gSpritesOnList
      puppetsprite (thisSprite, false)
    end repeat
    set gSpritesOnList = []
    repeat with N = FirstSprite to LastSprite
      puppetsprite N, true
      add gSpritesonList, N
    end repeat
  else
    set gSpritesOnList = []
    repeat with N = FirstSprite to LastSprite
      puppetsprite N, true
      add gSpritesonList, N
    end repeat
  end
end

```



```

end if
end

on Spritesoff FirstSprite, LastSprite
-- turns off sprites in consecutive channels from
-- FirstSprite to LastSprite
repeat with N = FirstSprite to LastSprite
  puppetsprite N, false
end repeat
end

```

Movie Script44:WaitHandlers

```

on wait Howlong
-- New Improved wait handler. doesn't reset timer
-- every time it's called. be sure to "StartTimer in
-- "on StartMovie"
set x = the timer
put x into oldtime -- stores time
repeat while (oldtime + Howlong) > x
  nothing
  set x = the timer
end repeat
end wait

on waitPlus Howlong, doWhat
-- Same as above handler except that allows passing
-- of a handler to be executed during the wait
-- Handler must be a string
set x = the timer
put x into oldtime -- stores time
repeat while (oldtime + Howlong) > x
  do doWhat -- must be a string
  set x = the timer
end repeat
end wait

on IgnoreMouseDowns
if the mousedown then dontpassevent
end

```

Score Script45

```
on exitFrame  
  cursor 4  
  displayRhymeTimedata  
  cursor -1  
end
```

Script of Cast Member48

```
on mouseUp  
  go to movie "dataTest"  
end
```

```

on initBalloonFields. -- empty Balloon's data fields of all but first Line
global gBalloonFields, gGamesViewedList, gTheFont

if not getOne (gGamesViewedList, #Balloo) then
  append gGamesViewedList, #Balloon
  repeat with thisfield in gBalloonFields
    Put " " into field thisfield
    set the font of field thisfield = gTheFont
    set the fontsize of field thisfield = 12
    set the fontStyle of line 1 of field thisfield = "underline"
    case (thisField) of
      "DataDate":Put "Date:" into text
      "BalloonTask":Put "Task :" into text
      "BalloonNumber":Put "Number:" into text
      "BalloonStimType":Put "Stimulus Type:" into text
      "BalloonVisDisplay":Put "Visual Display:" into text
      "BalloonNoise":Put "Noise:" into text
      "BalloonScore":Put "Cuml. Score:" into text
    end case
    put text into line 1 of field thisField
    set the fontStyle of line 1 of field thisfield = "underline"
    put " " INTO LINE 2 OF field thisField
    setPlainStyle(2,thisfield)
  end repeat
else
  repeat with thisfield in gBalloonFields
    set NumLines = the number of lines of field thisField
    if Line 2 of field thisfield = " " then exit repeat
    delete line 2 to numlines of field thisfield
    put " " INTO LINE 2 OF field thisField
    setPlainStyle(2,thisfield)
  end repeat
end if
end

```

Score Script62

```
on exitFrame
  cursor 4
  displayBalloonData
end
```

Score Script63

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalbuttonhandler() then
    cursor 4
    initCatConFields
    set gSessionNum = gSessionNum + 1
    newdisplayCatconData
    cursor -1
  end if
end
```

Score Script67

```
on mouseUp
  go to movie "dataTest"
end
```

Movie Script71

```
on MakeNameList x, y
  set nameList = []
  repeat with num = x to y
    set MemberName = the name of member num
    append nameList, membername
  end repeat
  put nameList
end
```

Score Script72

```
on exitFrame  
  cursor 4  
  NewdisplayCatConData  
  cursor -1  
end
```

```

on initCatConFields
-- empty "CataPillar Con"s data fields of all but first Line
global gcatConFields , gGamesViewedList, gTheFont
if not getOne(gGamesViewedList, #CatCon) then
    append gGamesViewedList, #CatCon
    repeat with thisfield in gcatConFields
        set NumLines = the number of lines of field thisfield
        put " " into field thisfield
        set the font of field thisfield = gTheFont
        set the fontsize of field thisfield = 12
        set the fontStyle of line 1 of field thisfield to "plain,underline"
        case (thisfield) of
            "DataDate":Put "Date:" into text
            "CatConTask":Put "Task :" into text
            "CatConUnits":Put "Units:" into text
            "CatConInterval":Put "Interval:" into text
            "CatConTarget":Put "Target:" into text
            "CatConFoils":Put "#Foils" into text
            "CatConScore":Put "Cuml. Score:" into text
        end case
        put text into line 1 of field thisfield
        set the fontStyle of line 1 of field thisfield to "plain,underline"
        delete line 2 to numlines of field thisfield
        put " " INTO LINE 2 OF field thisfield
        setPlainStyle(2,thisfield)
    end repeat
else
    repeat with thisfield in gcatConFields
        set NumLines = the number of lines of field thisfield
        if line 2 of field thisfield = " " then exit repeat
        delete line 2 to numlines of field thisfield
        put " " INTO LINE 2 OF field thisfield
        setPlainStyle(2,thisfield)
    end repeat
end if
end

```

Score Script74

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalbuttonHandler() then
    cursor 4
    initBalloonFields
    set gSessionNum = gSessionNum + 1
    displayBalloonData
    cursor -1
  end if
end
```

Score Script78

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalbuttonHandler() then
    spriteListOff
    updatestage
    -- initRtFields
    set gSessionNum = gSessionNum - 1
    go to Marker (0)
  end if
end
```

Score Script79

Score Script80

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalbuttonHandler() then
    spriteListOff
    updatestage
    -- initRtFields
    set gSessionNum = gSessionNum + 1
    go to Marker (0)
  end if
end
```

```

on displayRhymeTimeData
  global gRecordDisplay, gWhichUser, gSessionNum

  put getData (gRecordDisplay, gWhichUser, 3) into dataList
  set SessionNums = count(dataList)

  putUpNavArrows SessionNums

updatestage
  put getpropat(dataList, gSessionNum) into date
  put getProp(dataList,date) into sessionList
  put date into line 2 of field "dataDate"
  setPlainStyle(2,"dataDate")
  set NumRounds = count(sessionList)
  set Task1 = 0
  set task2 = 0
  repeat with x = 1 to numRounds
    put getpropat (sessionList,x ) into Level
    set LineNum1 = the number of lines of field "RT Score1"
    set LineNum2 = the number of lines of field "RT Score2"
    Case (Level) of
      "1","2","3","4","5": PutUpTask1Stats level
        put getProp (sessionList,level) & return into line linenum1 of field "RT Score1"
        setPlainStyle(linenum1,"RT score1")
        set task1 = 1
      "6","7","8","9","10","11": PutUpTask2Stats level
        put getProp (sessionList,level) & return into line linenum2 of field "RT Score2"
        setPlainStyle(linenum2,"RT score2")
        set task2 = 1
    end case
  end repeat
  if (task1 = 1) and (task2 = 1) then -- both tasks must be displayed
    set FieldSize = (the bottom of sprite 10) - (the top of sprite 10)
    repeat with x = 10 to 13
      set the loc of sprite (x+5) to (the loc of sprite x ) + point (0, fieldSize)
    end repeat
    updatestage
  end if
  if Task1 = 1 and task2 = 0 then -- just display task 1
    nothing
    exit
  end if
  if task1 = 0 and task2 = 1 then -- just display task 2
    repeat with x = 10 to 13
      set newCast = the member of sprite (x+5)
      set the member of sprite x to newCast
      set the loc of sprite (x + 5) to the loc of sprite x
    end repeat
    updatestage
  end if
end

```

```

on PutUpTask1Stats whichLine
  set whichLine = Value(whichLine)
  put field "RTLevelKeywords" into keywords
  set lineNum = the number of lines of field "RT Task(1)"
  put item 2 of line whichLine of keywords & return into line lineNum of field "RT
Task(1)"
  setPlainStyle(lineNum, "RT Task(1)")
  set lineNum = the number of lines of field "RT InSetOf"
  put item 3 of line whichLine of keywords & return into line lineNum of field "RT
InSetOf"
  setPlainStyle(lineNum, "RT InSetOf")
  set lineNum = the number of lines of field "RT BgNoise1"
  put item 4 of line whichLine of keywords & return into line lineNum of field "RT
BgNoise1"
  setPlainStyle(lineNum, "RT BgNoise1")
end

```

```

on PutUpTask2Stats whichLine
  set whichLine = Value(whichLine)
  put field "RTLevelKeywords" into keywords
  set lineNum = the number of lines of field "RT Task(2)"
  put item 2 of line whichLine of keywords & return into line lineNum of field "RT
Task(2)"
  setPlainStyle(lineNum, "RT Task(2)")
  put item 3 of line whichLine of keywords & return into line lineNum of field "RT
responseChoice"
  setPlainStyle(lineNum, "RT responseChoice")
  put item 4 of line whichLine of keywords & return into line lineNum of field "RT
BgNoise2"
  setPlainStyle(lineNum, "RT BgNoise2")
end

```

Score Script82

Movie Script83

```

on puppetButtonSprites whichIsIt
  -- turns on/off buttons for scrolling through sessions
  -- exclusive of spritesOnList handler
  global gNextButton, gPrevButton
  puppetsprite gNextButton, whichIsIt
  puppetsprite gPrevButton, whichIsIt
end

```

Score Script84

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalbuttonhandler() then
    cursor 4
    initCatConFields
    set gSessionNum = gSessionNum + 1
    newdisplayCatconData
    cursor -1
  end if
end
```

Score Script85

```
on exitFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  puppetButtonSprites true
  spritesonlist [28]
end
```

Score Script86

```
on exitFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  puppetButtonSprites true
  spritesonlist [28]
end
```

Score Script87

```
on mouseDown
    global gSessionNum
    if BlankArrow() then exit -- do nothing if blank arrow is up
    if legalbuttonHandler() then
        cursor 4
        initBalloonFields
        set gSessionNum = gSessionNum - 1
        displayBalloonData
        cursor -1
    end if
end
```

```

on PutUpEggTask2scores date, sessionList
  global gMoreButton, gMoreButtonLoc, gNextStart, gEgg2ob, gSessionNum, gOverRunSession
  global gNumPages

  set gOverRunSession = [:]

  if the machineType = 256 then
    set OverRunLine = 15
  else
    set OverRunLine = 17
  end if

  sendAwayViewPageButton
  updatestage

  set NumRounds = count(sessionList)

  if numRounds <= (OverRunLine + 1) then
    set aProp gOverRunSession, date, sessionList -- only need one sessionList
    set gNumPages = 1
  else
    if numRounds > (OverRunLine + 1) and numRounds <= ((2*OverRunLine) + 1) then
      -- split sessionList into two lists for each page of screen data
      set tempList1 = [:]
      set tempList2 = [:]
      set tempList3 = []
      repeat with x = 1 to overRunLine
        set aProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
      end repeat
      append tempList3, tempList1
      repeat with x = (overRunLine + 1) to numRounds
        set aProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
      end repeat
      append tempList3, tempList2
      set aProp gOverRunSession, date, tempList3
      set gNumPages = 2
    else
      if numRounds > ((2*OverRunLine) + 1) then
        -- need three screens to show more data
        set tempList1 = [:]
        set tempList2 = [:]
        set tempList3 = [:]
        set tempList4 = []
        repeat with x = 1 to overRunLine
          set aProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
        end repeat
        append tempList4, tempList1
        repeat with x = (overRunLine + 1) to overRunLine*2
          set aProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
        end repeat
        append tempList4, tempList2
        if numRounds <= (3*overRunLine) + 1 then
          -- there may be more than 3 pages worth of data so we
          -- need to pin out at three and allow for less than 3 as well
          set lastRound = numRounds
        end if
      end if
    end if
  end if
end on

```



```

else
    set lastRound = (3*overRunLine) + 1
end if
repeat with x = ((2*overRunLine) + 1) to lastRound
    setAProp tempList3, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append tempList4, tempList3
setaProp gOverRunSession, date, tempList4
set gNumPages = 3
end if
end if
end if

```

```

put date into line 2 of field fieldnum(gEgg2ob, #dataDate)
setPlainStyle(2, fieldnum(gEgg2ob, #dataDate))

```

```

case (gNumPages) of
    1: ShowEgg2Pagelof1
    2: ShowEgg2Pagelof2
    3: ShowEgg2Pagelof3
end case
end

```

```

on ShowEgg2Pagelof1
    global gOverRunSession, gEgg2ob
    set PageToShowList = getAt(gOverRunSession, 1)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getPropAt (PageToShowList, x) into Level
        set LineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
        PutUpEgg2words level
        put getProp (PageToShowList, level) into scoreList
        put getAt(scoreList, 1) & return into Line LineNum of field
        fieldNum(gEgg2ob, #Egg2DiffScore)
        setPlainStyle(lineNum, fieldNum(gEgg2ob, #Egg2DiffScore))
        put getAt(scoreList, 2) & return into Line LineNum of field
        fieldNum(gEgg2ob, #Egg2sameScore)
        setPlainStyle(lineNum, fieldNum(gEgg2ob, #Egg2sameScore))
    end repeat
end

```

```

on ShowEgg2Pagelof2
    global gOverRunSession, gPageNum, gEgg2ob
    sendAwayViewPageButton
    updatestage
    set date = getPropAt(gOverRunSession, 1)
    put date into line 2 of field fieldnum(gEgg2ob, #dataDate)
    setPlainStyle(2, fieldnum(gEgg2ob, #dataDate))
    putUpPglof2 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 1)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getPropAt (PageToShowList, x) into Level
        set LineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
        PutUpEgg2words level
        put getProp (PageToShowList, level) into scoreList
        put getAt(scoreList, 1) & return into Line LineNum of field
        fieldNum(gEgg2ob, #Egg2DiffScore)
    end repeat
end

```



```

    setPlainStyle(lineNum,fieldNum(gEgg2ob,#Egg2DiffScore))
    put getat(scoreList,2)&return into Line LineNum of field
fieldNum(gEgg2ob,#Egg2sameScore)
    setPlainStyle(lineNum,fieldNum(gEgg2ob,#Egg2sameScore))
end repeat
putUpEgg2Ellipses lineNum + 1

PutUpViewPage2of2 -- Button
set gPageNum = 1
end

on ShowEgg2Page2of2
global gOverRunSession, gPageNum, gEgg2ob
set date = getPropAt(gOverRunSession,1)
put date into line 2 of field fieldnum(gEgg2ob, #dataDate)
setPlainStyle(2,fieldnum(gEgg2ob, #dataDate))
putUpPg2of2 -- text prompt
set PagesToShowList = getAt(gOverRunSession, 1)
set PageToShowList = getAt (PagesToShowList, 2)
set numRounds = count(PageToShowList)
repeat with x = 1 to numRounds
    put getpropat (PageToShowList,x ) into Level
    set LineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
    PutUpEgg2words level
    put getProp (PageToShowList,level) into scoreList
    put getat(scoreList,1)&return into Line LineNum of field
fieldNum(gEgg2ob,#Egg2DiffScore)
    setPlainStyle(lineNum,fieldNum(gEgg2ob,#Egg2DiffScore))
    put getat(scoreList,2)&return into Line LineNum of field
fieldNum(gEgg2ob,#Egg2sameScore)
    setPlainStyle(lineNum,fieldNum(gEgg2ob,#Egg2sameScore))
end repeat

PutUpViewPagelof2 -- Button
set gPageNum = 2
end

on ShowEgg2Pagelof3
global gOverRunSession, gPageNum, gEgg2ob
sendAwayViewPageButton
updatestage
set date = getPropAt(gOverRunSession,1)
put date into line 2 of field fieldnum(gEgg2ob, #dataDate)
setPlainStyle(2,fieldnum(gEgg2ob, #dataDate))
putUpPglof3 -- text prompt
set PagesToShowList = getAt(gOverRunSession, 1)
set PageToShowList = getAt (PagesToShowList, 1)
set numRounds = count(PageToShowList)
repeat with x = 1 to numRounds
    put getpropat (PageToShowList,x ) into Level
    set LineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
    PutUpEgg2words level
    put getProp (PageToShowList,level) into scoreList
    put getat(scoreList,1)&return into Line LineNum of field
fieldNum(gEgg2ob,#Egg2DiffScore)
    setPlainStyle(lineNum,fieldNum(gEgg2ob,#Egg2DiffScore))
    put getat(scoreList,2)&return into Line LineNum of field
fieldNum(gEgg2ob,#Egg2sameScore)
    setPlainStyle(lineNum,fieldNum(gEgg2ob,#Egg2sameScore))
end repeat
putUpEgg2Ellipses lineNum + 1

```

```

PutUpViewPage2of3 -- Button
set gPageNum = 1
end

on ShowEgg2Page2of3
global gOverRunSession, gPageNum, gEgg2ob
set date = getPropAt(gOverRunSession, 1)
put date into line 2 of field fieldnum(gEgg2ob, #dataDate)
setPlainStyle(2, fieldnum(gEgg2ob, #dataDate))
putUpPg2of3 -- text prompt
set PagesToShowList = getAt(gOverRunSession, 1)
set PageToShowList = getAt (PagesToShowList, 2)
set numRounds = count(PageToShowList)
repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x ) into Level
    set LineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
    PutUpEgg2words level
    put getProp (PageToShowList, level) into scoreList
    put getat(scoreList, 1) & return into Line LineNum of field
fieldNum(gEgg2ob, #Egg2DiffScore)
    setPlainStyle(lineNum, fieldNum(gEgg2ob, #Egg2DiffScore))
    put getat(scoreList, 2) & return into Line LineNum of field
fieldNum(gEgg2ob, #Egg2sameScore)
    setPlainStyle(lineNum, fieldNum(gEgg2ob, #Egg2sameScore))
end repeat
putUpEgg2Ellipses lineNum + 1

PutUpViewPage3of3 -- Button
set gPageNum = 2
end

on ShowEgg2Page3of3
global gOverRunSession, gPageNum, gEgg2ob
set date = getPropAt(gOverRunSession, 1)
put date into line 2 of field fieldnum(gEgg2ob, #dataDate)
setPlainStyle(2, fieldnum(gEgg2ob, #dataDate))
putUpPg3of3 -- text prompt
set PagesToShowList = getAt(gOverRunSession, 1)
set PageToShowList = getAt (PagesToShowList, 3)
set numRounds = count(PageToShowList)
repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x ) into Level
    set LineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
    PutUpEgg2words level
    put getProp (PageToShowList, level) into scoreList
    put getat(scoreList, 1) & return into Line LineNum of field
fieldNum(gEgg2ob, #Egg2DiffScore)
    setPlainStyle(lineNum, fieldNum(gEgg2ob, #Egg2DiffScore))
    put getat(scoreList, 2) & return into Line LineNum of field
fieldNum(gEgg2ob, #Egg2sameScore)
    setPlainStyle(lineNum, fieldNum(gEgg2ob, #Egg2sameScore))
end repeat

PutUpViewPage1of3 -- Button
set gPageNum = 3
end

```

```

on PutUpEgg2words whichLine
  global gEgg2ob
  set WhichLine = value(whichLine)
  put field "EggLevelKeywords" into keywords
  set lineNum = the number of lines of field fieldnum(gegg2ob, #EggTask2)
  put item 1 of line whichLine of keywords & return into line lineNum of field
  fieldnum(gegg2ob, #EggTask2)
  setPlainStyle(lineNum, fieldnum(gegg2ob, #EggTask2))
  set lineNum = the number of lines of field fieldnum(gEgg2ob, #EggDuration)
  put item 2 of line whichLine of keywords & return into line lineNum of field
  fieldnum(gEgg2ob, #EggDuration)
  setPlainStyle(lineNum, fieldnum(gEgg2ob, #EggDuration))
  set lineNum = the number of lines of field fieldnum(gEgg2ob, #EggAmplification)
  put item 3 of line whichLine of keywords & return into line lineNum of field
  fieldnum(gEgg2ob, #EggAmplification)
  setPlainStyle(lineNum, fieldnum(gEgg2ob, #EggAmplification))
  set lineNum = the number of lines of field fieldnum(gEgg2ob, #EggSteps)
  put item 4 of line whichLine of keywords & return into line lineNum of field
  fieldnum(gEgg2ob, #EggSteps)
  setPlainStyle(lineNum, fieldnum(gEgg2ob, #EggSteps))
end

on putUpEgg2Ellipses whichLine
  global gEggTask2Fields
  repeat with x in gEggTask2Fields
    if x = "DataDate" then next repeat
    put " " into line whichLine of field x
    -- set the textstyle of line whichLine of field x to "italic,underline"

    put "...more..." into line whichLine of field x
    setPlainStyle(whichLine, x)
  end repeat
end

```

Movie Script89:newCoal1Display

```
on PutUpCoal1Data
  global gRecordDisplay, gWhichUser, gSessionNum, gNextStart, gOverRunSession, gNumPages
  Global gPageNum

  -- 5/1/97
  -- shows up to three screens of data for any session
  -- completely revised data display procedures.

  set gNumPages = 0
  set gPageNum = 0

  if the machineType = 255 then
    set OverRunLine = 19
  else
    set OverRunLine = 22
  end if

  put getData (gRecordDisplay, gWhichUser, 6) into dataList

  set SessionNums = count (dataList)

  putUpNavArrows sessionNums

  updatestage

  set gOverRunSession = [:]

  put getpropat (dataList, gSessionNum) into date
  put getat (dataList, gSessionNum) into SessionList

  set NumRounds = count (sessionList)

  -- Here we need to look at numRounds and decide how many pages we will
  -- need to show the data. If it's one then we're good, if two or three
  -- then we need to break sessionList into pieces to feed into the
  -- display routines, and store those pieces as a list of lists
  -- in a property list with "date" as property

  if numRounds <= (OverRunLine + 1) then
    set aProp gOverRunSession, date, sessionList -- only need one sessionList
    set gNumPages = 1
  else
    if numRounds > (OverRunLine + 1) and numRounds <= ((2*OverRunLine) + 1) then
      -- split sessionList into two lists for each page of screen data
      set tempList1 = [:]
      set tempList2 = [:]
      set tempList3 = []
      repeat with x = 1 to overRunLine
        setAProp tempList1, getPropAt (sessionList, x), getAt (sessionList, x)
      end repeat
      append tempList3, tempList1
      repeat with x = (overRunLine + 1) to numRounds
        setAProp tempList2, getPropAt (sessionList, x), getAt (sessionList, x)
      end repeat
    end if
  end if
```

```

append tempList3, tempList2
setaprop gOverRunSession, date, tempList3
set numPages = 2
else
  if (2*overRunLine) < 10 then
    need three screens to show all the data
    set tempList1 = {}
    set tempList2 = {}
    set tempList3 = {}
    set tempList4 = {}
    repeat with x = 1 to overRunLine
      setAProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
    end repeat
    append tempList4, tempList1
    repeat with x = overRunLine + 1 to overRunLine*2
      setAProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
    end repeat
    append tempList4, tempList2
    repeat with x = (2*overRunLine) + 1 to numRounds
      setAProp tempList3, getPropAt(sessionList, x), getAt(sessionList, x)
    end repeat
    append tempList4, tempList3
    setaprop gOverRunSession, date, tempList4
    set numPages = 3
  end if
end if
end if
end if

```

```

-- line 2 of field "dataDate"
-- line 2 of field "dataDate"
-- away from Page-Button
-- dataDate

```

```

-- numPages = 3
-- away from Page-Button
-- away from Page-Button
-- away from Page-Button
end if
end

```

```

on ShowCCPage1of1
  global gOverRunSession
  set PageToShowList = getAt(gOverRunSession, 1)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x) into Level
    set LineNum = the number of lines of field "Coaltask1"
    PutUpCoallwords level
    put getProp (PageToShowList, level) & return into line linenum of field "Coallscore"
    setPlainStyle(lineNum, "Coallscore")
  end repeat
end

```

```

on ShowCCPage1of2
  global gOverRunSession, gPageNum

```



```

set date = getPropAt(gOverRunSession,1)
put date into line 2 of field "dataDate"
setPlainStyle(2,"dataDate")
putUpPglof2 -- text prompt
set PageToShowList = getAt(gOverRunSession, 1)
set PageToShowList = getAt (PageToShowList, 1)
set numRounds = count(PageToShowList)
repeat with x = 1 to numRounds
    put getpropat (PageToShowList,x ) into Level
    set LineNum = the number of lines of field "Coaltask1"
    PutUpCoallwords level
    put getProp (PageToShowList,level) & return into line linenum of field "Coallscore"
    setPlainStyle(lineNum,"Coallscore")
end repeat
putUpCoalEllipses lineNum + 1
PutUpViewPage1of2 -- Button

set gPageNum = 1

on ShowCCPage2of2
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession,1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2,"dataDate")
    putUpPglof2 -- text prompt
    set PageToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PageToShowList, 1)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList,x ) into Level
        set LineNum = the number of lines of field "Coaltask1"
        PutUpCoallwords level
        put getProp (PageToShowList,level) & return into line linenum of field "Coallscore"
        setPlainStyle(lineNum,"Coallscore")
    end repeat
    PutUpViewPage2of2 -- Button

set gPageNum = 2
end

on ShowCCPage3of3
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession,1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2,"dataDate")
    putUpPglof3 -- text prompt
    set PageToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PageToShowList, 1)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList,x ) into Level
        set LineNum = the number of lines of field "Coaltask1"
        PutUpCoallwords level
        put getProp (PageToShowList,level) & return into line linenum of field "Coallscore"
        setPlainStyle(lineNum,"Coallscore")
    end repeat
    putUpCoalEllipses lineNum + 1
    PutUpViewPage2of3 -- Button

set gPageNum = 1

```

```

end

on ShowCCPage2of3
  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession,1)
  put date into line 2 of field "dataDate"
  setPlainStyle(2,"dataDate")
  putUpPg2of3 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 2)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList,x ) into Level
    set LineNum = the number of lines of field "Coaltask1"
    PutUpCoaliwords level
    put getProp (PageToShowList,level) & return into line linenum of field "Coallscore"
    setPlainStyle(lineNum,"Coallscore")
  end repeat
  putUpCoaliwords lineNum + 1
  PutUpViewPage3of3 -- Button

  set gPageNum = 2
end

```

```

on ShowCCPage3of3
  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession,1)
  put date into line 2 of field "dataDate"
  setPlainStyle(2,"dataDate")
  putUpPg3of3 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 3)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList,x ) into Level
    set LineNum = the number of lines of field "Coaltask1"
    PutUpCoaliwords level
    put getProp (PageToShowList,level) & return into line linenum of field "Coallscore"
    setPlainStyle(lineNum,"Coallscore")
  end repeat
  PutUpViewPage1of3 -- Button

  set gPageNum = 3
end

```

```

on PutUpCoaliwords whichLine
  set WhichLine = value(whichLine)
  put field "CCKeywords" into keywords
  set lineNum = the number of lines of field "Coaltask1"
  put item 1 of line whichLine of keywords & return into line lineNum of field
  "Coaltask1"
  setPlainStyle (lineNum,"Coaltask1")
  set lineNum = the number of lines of field "CoallContext"
  put item 2 of line whichLine of keywords & return into line lineNum of field
  "CoallContext"
  setPlainStyle (lineNum,"CoallContext")

```


end

```
on putUpCoalEllipses whichLine
  global gcoalFields
  repeat with x in gcoalFields
    if x = "DataDate" then next repeat
    put " " into line whichLine of field x
    put "...more..." into line whichLine of field x
    setPlainStyle (whichLine,x )
  end repeat
end
```

Movie Script90:DisplayEggData

```
on DisplayEggData
  global gRecordDisplay, gWhichUser, gSessionNum, gNextStart, gMoreButton, gMoreButtonLoc
  put getEggBasketData (gRecordDisplay, gWhichUser) into dataList

  set SessionNums = count(dataList)
  putUpNavArrows SessionNums

  updatestage

  put getpropat(dataList, gSessionNum) into date
  put getat(dataList, gSessionNum) into SessionList
  -- here we need to see if the scores are all in the vowel game
  -- ie task 1: all in the CV game ie task 2 or in a mix of the two games
  -- so we need to scan the properties in sessionlist and check for values
  -- above and below 30 - the last vowel game
  set levelList = {}
  set numRounds = count(sessionList)
  repeat with x = 1 to numRounds
    append levelList, getPropat(sessionList, x)
  end repeat

  if max(levelList) < 31 then -- all vowels
    go to frame "gTask1"
    PutUpEggTaskScores (date, sessionList)
  end if
  if min(levelList) > 30 then -- all CV
    go to frame "gTask2"
    PutUpEggTaskScores (date, sessionList)
  end if
end

on PutUpEggTaskScores date, sessionList
  global gMoreButton, gMoreButtonLoc, gNextStart, gSessionNum, gOverRunSession, gNumPages
  global gPageNum

  set gOverRunSession = {}
  set gNumPages = 1
  set gPageNum = 1

  if the machineType = 256 then
    set OverRunLine = 17
  else
    set overRunLine = 19
  end if

  set NumRounds = count(sessionList)
  sendAwayViewPageButton
```

```

if numRounds <= (OverRunLine + 1) then
  setaprop gOverRunSession, date, sessionList -- only need one sessionList
  set gNumPages = 1
else -- break sessionList into 2 lists.
  set tempList1 = [:]
  set tempList2 = [:]
  set tempList3 = []
  repeat with x = 1 to overRunLine
    setAProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
  end repeat
  append tempList3, tempList1
  repeat with x = (overRunLine + 1) to numRounds
    setAProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
  end repeat
  append tempList3, tempList2
  setaprop gOverRunSession, date, tempList3
  set gNumPages = 2
end if

```

```

put date into line 2 of field "dataDate"
setPlainStyle(2, "dataDate")

```

```

case (gNumPages) of
  1: ShowEgglPagelof1
  2: ShowEgglPagelof2
end case
end

```

```

on ShowEgglPagelof1

```

```

  global gOverRunSession
  set PageToShowList = getAt(gOverRunSession, 1)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x) into Level
    set LineNum = the number of lines of field "Eggvowels"
    PutUpEgglwords level
    put getProp (PageToShowList, level) into scoreList
    put getat(scoreList, 1)&return into Line LineNum of field "EgglDiffScore"
    setPlainStyle(lineNum, "EgglDiffScore")
    put getat(scoreList, 2)&return into Line LineNum of field "EgglSameScore"
    setPlainStyle(lineNum, "EgglSameScore")
  end repeat
end

```

```

on ShowEgglPagelof2

```

```

  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession, 1)
  put date into line 2 of field "dataDate"
  setPlainStyle(2, "dataDate")
  putUpPglof2 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 1)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x) into Level
    set LineNum = the number of lines of field "Eggvowels"
    PutUpEgglwords level
    put getProp (PageToShowList, level) into scoreList
    put getat(scoreList, 1)&return into Line LineNum of field "EgglDiffScore"
  end repeat
end

```

```

    setPlainStyle(lineNum, "Egg1DiffScore")
    put getat(scoreList, 2) & return into Line LineNum of field "Egg1SameScore"
    setPlainStyle(lineNum, "Egg1SameScore")
end repeat
putUpEgg1Ellipses NumRounds + 2
PutUpViewPage2of2 -- Button

set gPageNum = 1
end

on ShowEgg1Page2of2
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession, 1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2, "dataDate")
    putUpPg2of2 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 2)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList, x) into Level
        set LineNum = the number of lines of field "Eggvowels"
        PutUpEgg1words level
        put getProp (PageToShowList, level) into scoreList
        put getat(scoreList, 1) & return into Line LineNum of field "Egg1DiffScore"
        setPlainStyle(lineNum, "Egg1DiffScore")
        put getat(scoreList, 2) & return into Line LineNum of field "Egg1SameScore"
        setPlainStyle(lineNum, "Egg1SameScore")
    end repeat
    PutUpViewPage1of2 -- Button

    set gPageNum = 2
end

on PutUpEgg1words whichLine
    set WhichLine = value(whichLine)
    put field "EggLevelKeywords" into keywords
    set lineNum = the number of lines of field "EggTask1"
    put item 1 of line whichLine of keywords & return into line lineNum of field "EggTask1"
    setPlainStyle(lineNum, "EggTask1")
    set lineNum = the number of lines of field "EggVowels"
    put item 2 of line whichLine of keywords & return into line lineNum of field "EggVowels"
    setPlainStyle(lineNum, "EggVowels")
end

on putUpEgg1Ellipses whichLine
    global gEggTask1Fields
    repeat with x in gEggTask1Fields
        if x = "DataDate" then next repeat
        put " " into line whichLine of field x
        put "...more..." into line whichLine of field x
        setPlainStyle(whichLine, X)
    end repeat
end

```

Movie Script91:NewDisplayCatConData

```
on NewDisplayCatConData
  global gRecordDisplay, gWhichUser, gSessionNum, gNextStart, gNumPages,
  gPageNum, gOverRunSession

  set gNumPages = 0
  set gPageNum = 0
  set gOverRunSession = [:]

  put getData (gRecordDisplay, gWhichUser, 1) into dataList

  set SessionNums = count(dataList)

  putUpNavArrows SessionNums

  if the machineType = 256 then
    set overRunLine = 19
  else
    set overRunLine = 22
  end if

  updatestage
  put getpropat(dataList, gSessionNum) into date
  put getProp(dataList, date) into SessionList

  set NumRounds = count(sessionList)

  -- Here we need to look at numRounds and decide how many pages we will
  -- need to show the data. If it's one then we're good, if two or three
  -- then we need to break sessionList into pieces to feed into the
  -- display routines, and store those pieces as a list of lists
  -- in a property list with "date" as property

  if numRounds <= (OverRunLine + 1) then
    setaprop gOverRunSession, date, sessionList -- only need one sessionList
    set gNumPages = 1
  else
    if numRounds > (OverRunLine + 1) and numRounds <= ((2*OverRunLine) + 1) then
      -- split sessionList into two lists for each page of screen data
      set tempList1 = [:]
      set tempList2 = [:]
      set tempList3 = []
      repeat with x = 1 to overRunLine
        setAProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
      end repeat
      append tempList3, tempList1
      repeat with x = (overRunLine + 1) to numRounds
        setAProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
      end repeat
      append tempList3, tempList2
      setaprop gOverRunSession, date, tempList3
      set gNumPages = 2
    else
      if numRounds > ((2*OverRunLine) + 1) then
        -- need three screens to show all the data
        set tempList1 = [:]
        set tempList2 = [:]
```



```

set templist3 = []
set templist4 = []
repeat with x = 1 to overRunLine
  setAProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append templist4, tempList1
repeat with x = (overRunLine + 1) to overRunLine*2
  setAProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append templist4, tempList2
repeat with x = ((2*overRunLine) + 1) to numrounds
  setAProp tempList3, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append templist4, tempList3
setaprop gOverRunSession, date, tempList4
set gNumPages = 3
end if
end if
end if

```

```

put date into line 2 of field "dataDate"
setPlainstyle(2, "dataDate")
sendAwayViewPageButton

```

```

case (gNumPages) of
  1: ShowCatConPagelof1
  2: ShowCatConPagelof2
  3: ShowCatConPagelof3
end case
end

```

```

on ShowCatConPagelof1
  global gOverRunSession
  set PageToShowList = getAt(gOverRunSession, 1)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x) into Level
    set LineNum = the number of lines of field "CatconScore"
    PutUpCatConwords level
    put getProp (PageToShowList, level) & return into line linenum of field "CatConScore"
    setPlainstyle(lineNum, "CatConScore")
  end repeat
end

```

```

on ShowCatConPagelof2
  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession, 1)
  put date into line 2 of field "dataDate"
  setPlainstyle(2, "dataDate")
  putUpPglof2 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 1)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x) into Level
    set LineNum = the number of lines of field "CatconScore"
    PutUpCatConwords level
    put getProp (PageToShowList, level) & return into line linenum of field "CatConScore"
  end repeat
end

```

```

    setPlainStyle(lineNum, "CatConScore")
end repeat
putUpCatConEllipses lineNum + 1
PutUpViewPage2of2 -- Button
set gPageNum = 1
end

on ShowCatConPage2of2
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession, 1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2, "dataDate")
    putUpPg2of2 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 2)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList, x) into Level
        set LineNum = the number of lines of field "CatConScore"
        PutUpCatConwords level
        put getProp (PageToShowList, level) & return into line linenum of field "CatConScore"
        setPlainStyle(lineNum, "CatConScore")
    end repeat
    PutUpViewPage1of2 -- Button
    set gPageNum = 2
end

on ShowCatConPage1of3
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession, 1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2, "dataDate")
    putUpPg1of3 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 1)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList, x) into Level
        set LineNum = the number of lines of field "CatConScore"
        PutUpCatConwords level
        put getProp (PageToShowList, level) & return into line linenum of field "CatConScore"
        setPlainStyle(lineNum, "CatConScore")
    end repeat
    putUpCatConEllipses lineNum + 1
    PutUpViewPage2of3 -- Button
    set gPageNum = 1
end

on ShowCatConPage2of3
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession, 1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2, "dataDate")
    putUpPg2of3 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 2)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList, x) into Level
        set LineNum = the number of lines of field "CatConScore"
        PutUpCatConwords level

```



```

    put getProp (PageToShowList,level) & return into line linenum of field "CatConScore"
    setPlainStyle(lineNum,"CatConScore")
end repeat
putUpCatConEllipses lineNum + 1
PutUpViewPage3of3 -- Button
set gPageNum = 2
end

```

```

on ShowCatConPage3of3
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession,1)
    put date into line 2 of field "dataDate"
    setPlainStyle(2,"dataDate")
    putUpPg3of3 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 3)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getpropat (PageToShowList,x ) into Level
        set LineNum = the number of lines of field "CatconScore"
        PutUpCatConwords level
        put getProp (PageToShowList,level) & return into line linenum of field "CatConScore"
        setPlainStyle(lineNum,"CatConScore")
    end repeat
    PutUpViewPage1of3 -- Button
    set gPageNum = 3
end

```

```

on PutUpCatConwords whichLine
    set WhichLine = value(whichLine)
    put field "CatConKeywords" into keywords
    set lineNum = the number of lines of field "CatConTask"
    put item 1 of line whichLine of keywords & return into line lineNum of field
    "CatConTask"
    setPlainStyle(lineNum,"CatConTask")
    set lineNum = the number of lines of field "CatConUnits"
    put item 2 of line whichLine of keywords & return into line lineNum of field
    "CatConUnits"
    setPlainStyle(lineNum,"CatConUnits")
    set lineNum = the number of lines of field "CatConInterval"
    put item 3 of line whichLine of keywords&"sec"& return into line lineNum of field
    "CatConInterval"
    setPlainStyle(lineNum,"CatConInterval")
    set lineNum = the number of lines of field "CatConTarget"
    put item 4 of line whichLine of keywords & return into line lineNum of field
    "CatConTarget"
    setPlainStyle(lineNum,"CatConTarget")
    set lineNum = the number of lines of field "CatConFoils"
    put item 5 of line whichLine of keywords & return into line lineNum of field
    "CatConFoils"
    setPlainStyle(lineNum,"CatConFoils")
end

```

```

on putUpCatConEllipses whichLine
    global gCatConFields

```

```

repeat with x in gCatConFields
  if x = "DataDate" then next repeat
  put " " into line whichLine of field x

  if x = "catConscore" then
    put "...more..." into line whichLine of field x
  else
    put "...more..." into line whichLine of field x
  end if
  setPlainStyle(whichLine.x)
end repeat
end

```

```

on DisplayBalloonData
  global gRecordDisplay, gWhichUser, gSessionNum, gNextStart, gBalloonfields, gBlinOb
  global gOverRunSession, gNumPages, gPageNum

  set gOverRunSession = [:]
  set gNumPages = 1
  set gPageNum = 1
  put getData (gRecordDisplay, gWhichUser, 5) into dataList -- retrieve initial data

  set SessionNums = count(dataList)

  putUpNavArrows SessionNums

  if the machineType = 256 then
    set overRunLine = 17
  else
    set overRunLine = 20
  -- set overRunLine = 17 -- testing! testing! change back!
  end if

  sendAwayViewPageButton

  put getpropat(dataList, gSessionNum) into date
  put getProp(dataList, date) into sessionList

  set NumRounds = count(sessionList)

  -- Here we need to look at numRounds and decide how many pages we will
  -- need to show the data. If it's one then we're good, if two or three
  -- then we need to break sessionList into pieces to feed into the
  -- display routines, and store those pieces as a list of lists
  -- in a property list with "date" as property

  if numRounds <= (OverRunLine + 1) then
    setaprop gOverRunSession, date, sessionList -- only need one sessionList
    set gNumPages = 1
  else
    if numRounds > (OverRunLine + 1) and numRounds <= ((2*OverRunLine) + 1) then
      -- split sessionList into two lists for each page of screen data
      set tempList1 = [:]
      set tempList2 = [:]
      set tempList3 = []
      repeat with x = 1 to overRunLine
        setAProp tempList1, getPropat(sessionList, x), getAt(sessionList, x)
      end repeat
      append tempList3, tempList1
      repeat with x = (overRunLine + 1) to numRounds
        setAProp tempList2, getPropat(sessionList, x), getAt(sessionList, x)
      end repeat
      append tempList3, tempList2
      setaprop gOverRunSession, date, tempList3
      set gNumPages = 2
    else
      if numRounds > ((2*OverRunLine) + 1) then
        -- need three screens to show all the data
        set tempList1 = [:]
        set tempList2 = [:]
        set tempList3 = [:]

```

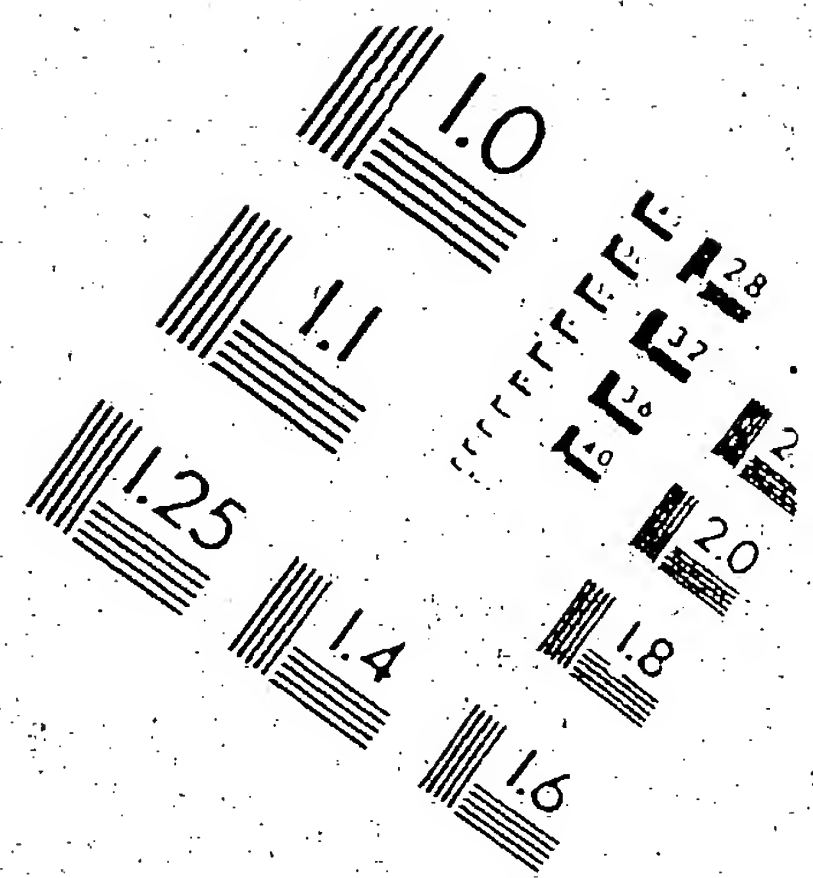
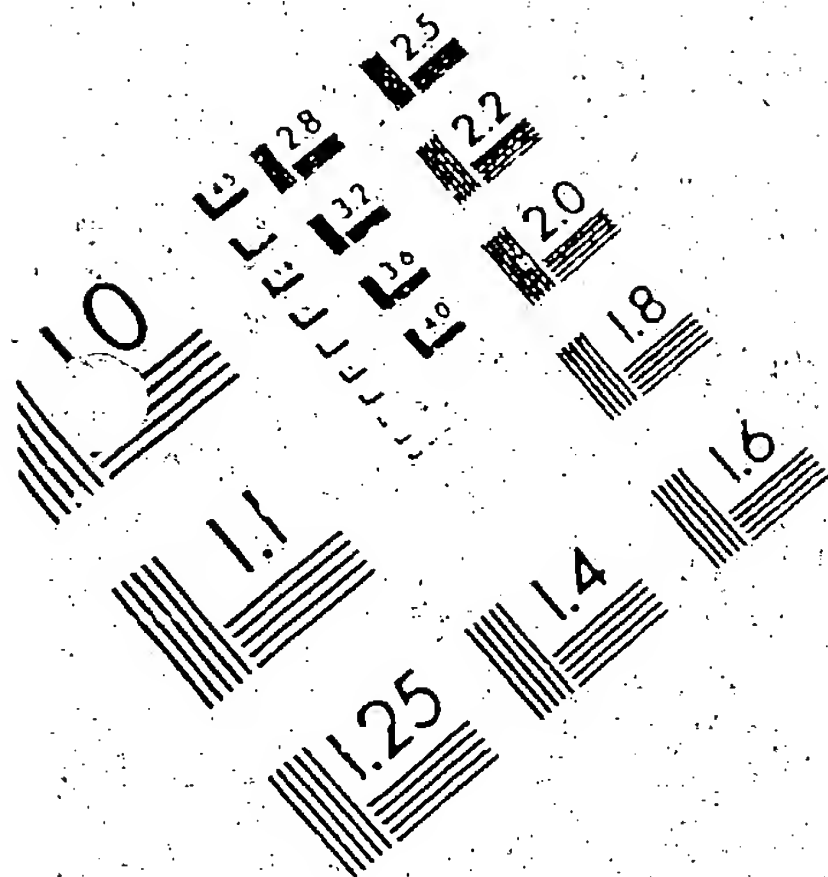
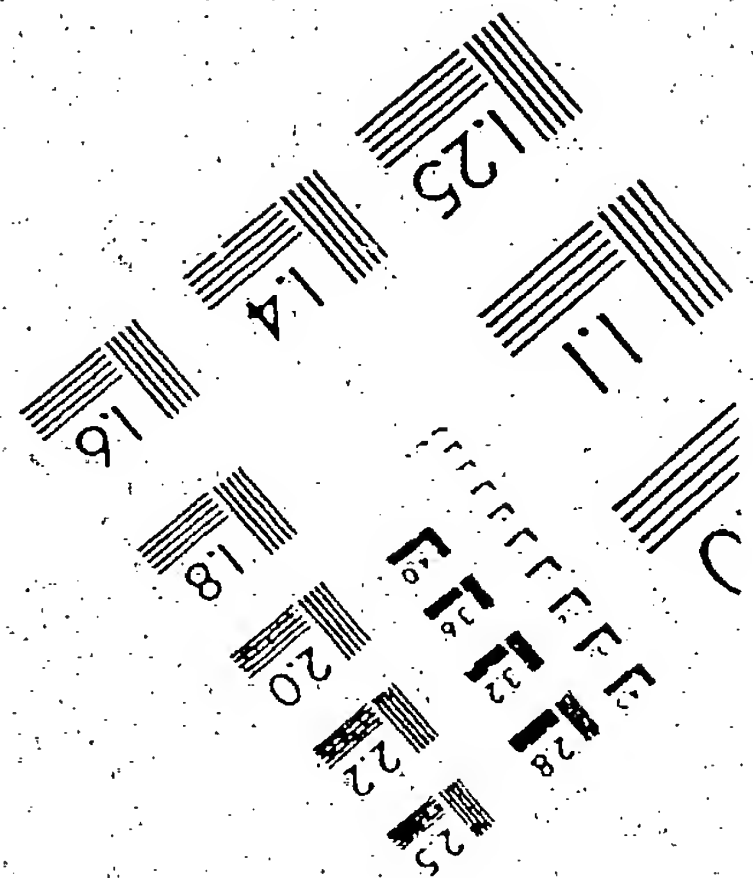
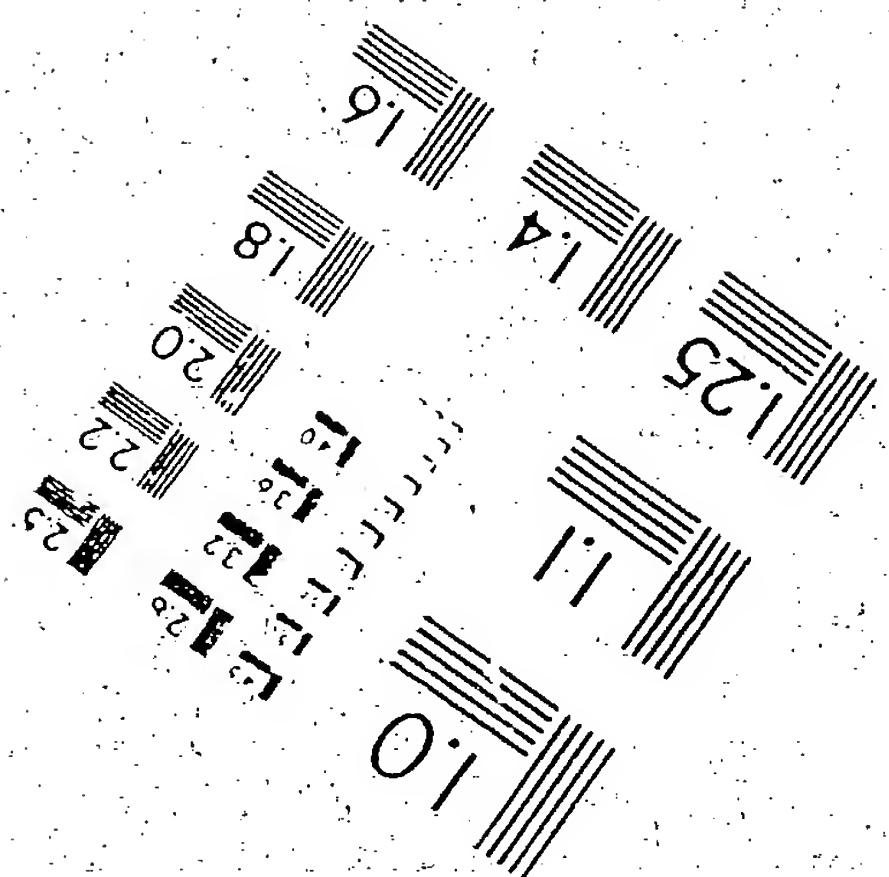
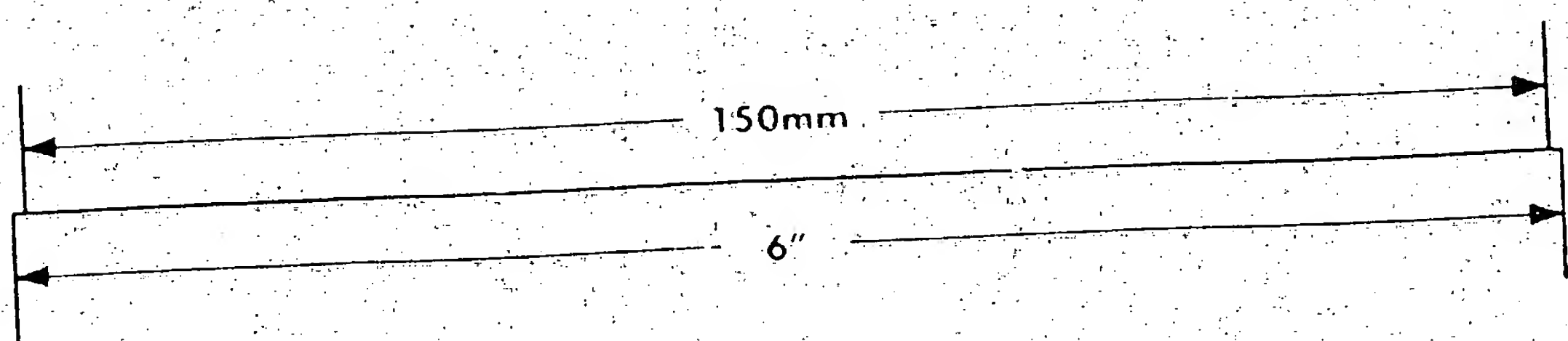
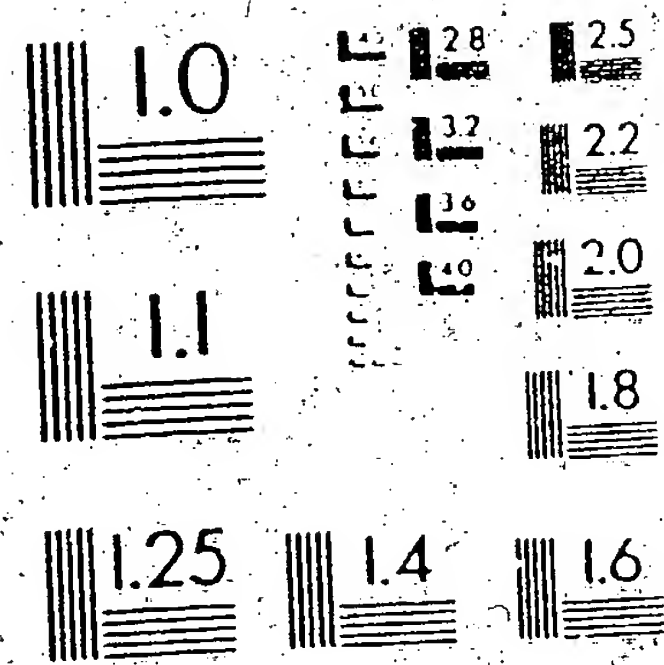


IMAGE EVALUATION TEST TARGET QA-3



APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone 716-482-0300
Fax 716-288-5989

```

set tempList4 = {}
repeat with x = 1 to overRunLine
    setAProp tempList1, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append tempList4, tempList1
repeat with x = (overRunLine + 1) to overRunLine*2
    setAProp tempList2, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append tempList4, tempList2
repeat with x = ((2*overRunLine) + 1) to numRounds
    setAProp tempList3, getPropAt(sessionList, x), getAt(sessionList, x)
end repeat
append tempList4, tempList3
setAProp gOverRunSession, date, tempList4
set gNumPages = 3
end if
end if
end if

put date into line 2 of field fieldNum(gBllnob, #dataDate)
setPlainStyle(2, fieldNum(gBllnob, #dataDate))

case (gNumPages) of
    1: ShowBalloonPagelof1
    2: ShowBalloonPagelof2
    3: ShowBalloonPagelof3
end case
end

on ShowBalloonPagelof1
    global gOverRunSession
    set PageToShowList = getAt(gOverRunSession, 1)
    set numRounds = count(PageToShowList)
    repeat with x = 1 to numRounds
        put getPropat (PageToShowList, x) into level
        PutUpBalloonwords (level, X + 1) -- 2nd param is what lineNum to put text on
        put getProp (PageToShowList, level) & return into line x + 1 of field
        fieldNum(gBllnob, #BalloonScore)
        setPlainStyle(x + 1, fieldNum(gBllnob, #BalloonScore))
    end repeat
end

on ShowBalloonPagelof2
    global gOverRunSession, gPageNum
    set date = getPropAt(gOverRunSession, 1)
    put date into line 2 of field fieldNum(gBllnob, #dataDate)
    setPlainStyle(2, fieldNum(gBllnob, #dataDate))
    putUpPglof2 -- text prompt
    set PagesToShowList = getAt(gOverRunSession, 1)
    set PageToShowList = getAt (PagesToShowList, 1)
    set numRounds = count (PageToShowList)
    repeat with x = 1 to numRounds
        put getPropat (PageToShowList, x) into level
        PutUpBalloonwords (level, X + 1) -- 2nd param is what lineNum to put text on
        put getProp (PageToShowList, level) & return into line x + 1 of field
        fieldNum(gBllnob, #BalloonScore)
        setPlainStyle(x + 1, fieldNum(gBllnob, #BalloonScore))
    end repeat
    putUpBalloonEllipses NumRounds + 2
    PutUpViewPage2of2 -- Button

```



```

set gPageNum = 1
end

on ShowBalloonPage1of3
  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession, 1)
  put date into line 2 of field fieldNum(gBllnob, #dataDate)
  setPlainStyle(2, fieldNum(gBllnob, #dataDate))
  putUpPg1of3 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 1)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x ) into Level
    PutUpBalloonwords (level, X + 1 ) -- 2nd param is what lineNum to put text on
    put getProp (PageToShowList, level) & return into line x +1 of field
    fieldNum(gBllnob, #BalloonScore)
    setPlainStyle(x +1, fieldNum(gBllnob, #BalloonScore))
  end repeat
  putUpBalloonEllipses NumRounds + 2
  PutUpViewPage2of3 -- Button

  set gPageNum = 1
end

on ShowBalloonPage2of2
  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession, 1)
  put date into line 2 of field fieldNum(gBllnob, #dataDate)
  setPlainStyle(2, fieldNum(gBllnob, #dataDate))
  putUpPg2of2 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 2)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x ) into Level
    PutUpBalloonwords (level, X + 1 ) -- 2nd param is what lineNum to put text on
    put getProp (PageToShowList, level) & return into line x +1 of field
    fieldNum(gBllnob, #BalloonScore)
    setPlainStyle(x +1, fieldNum(gBllnob, #BalloonScore))
  end repeat
  PutUpViewPage1of2 -- Button

  set gPageNum = 2
end

on ShowBalloonPage2of3
  global gOverRunSession, gPageNum
  set date = getPropAt(gOverRunSession, 1)
  put date into line 2 of field fieldNum(gBllnob, #dataDate)
  setPlainStyle(2, fieldNum(gBllnob, #dataDate))
  putUpPg2of3 -- text prompt
  set PagesToShowList = getAt(gOverRunSession, 1)
  set PageToShowList = getAt (PagesToShowList, 2)
  set numRounds = count(PageToShowList)
  repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x ) into Level
    PutUpBalloonwords (level, X + 1 ) -- 2nd param is what lineNum to put text on
    put getProp (PageToShowList, level) & return into line x +1 of field
    fieldNum(gBllnob, #BalloonScore)
    setPlainStyle(x +1, fieldNum(gBllnob, #BalloonScore))

```

```

end repeat
putUpBalloonEllipses NumRounds + 2
PutUpViewPage3of3 -- Button

set gPageNum = 2
end

on ShowBalloonPage3of3
global gOverRunSession, gPageNum
set date = getPropAt(gOverRunSession, 1)
put date into line 2 of field fieldNum(gBllnob, #dataDate)
setPlainStyle(2, fieldNum(gBllnob, #dataDate))
putUpPg3of3 -- text prompt
set PagesToShowList = getAt(gOverRunSession, 1)
set PageToShowList = getAt(PagesToShowList, 3)
set numRounds = count(PageToShowList)
repeat with x = 1 to numRounds
    put getpropat (PageToShowList, x) into Level
    PutUpBalloonwords (level, x + 1) -- 2nd param is what lineNum to put text on
    put getProp (PageToShowList, level) & return into line x + 1 of field
fieldNum(gBllnob, #BalloonScore)
    setPlainStyle(x + 1, fieldNum(gBllnob, #BalloonScore))
end repeat
PutUpViewPage1of3 -- Button

set gPageNum = 3
end

on PutUpBalloonwords whichLine, lineNum
set WhichLine = value(whichLine)
put field "BalloonKeywords" into keywords
put item 1 of line whichLine of keywords & return into line lineNum of field
fieldNum(gBllnob, #BalloonTask)
setPlainStyle(lineNum, fieldNum(gBllnob, #BalloonTask))
put item 2 of line whichLine of keywords & return into line lineNum of field
fieldNum(gBllnob, #BalloonNumber)
setPlainStyle(lineNum, fieldNum(gBllnob, #BalloonNumber))
put item 3 of line whichLine of keywords & return into line lineNum of field
fieldNum(gBllnob, #BalloonStimType)
setPlainStyle(lineNum, fieldNum(gBllnob, #BalloonStimType))
put item 4 of line whichLine of keywords & return into line lineNum of field
fieldNum(gBllnob, #BalloonVisDisplay)
setPlainStyle(lineNum, fieldNum(gBllnob, #BalloonVisDisplay))
put item 5 of line whichLine of keywords & return into line lineNum of field
fieldNum(gBllnob, #BalloonNoise)
setPlainStyle(lineNum, fieldNum(gBllnob, #BalloonNoise))
end

on putUpBalloonEllipses whichLine
global gBalloonfields
repeat with x in gBalloonfields
    if x = "dataDate" then next repeat
    put " " into line whichLine of field x
    put "...more..." in line whichLine of field x
    setPlainStyle(whichLine, x)
end repeat
end

```


Score Script93

```
on mouseDown
  global gPageNum, gNumPages
  if legalButtonHandler() then
    cursor 4
    if gPageNum = 1 then
      initegg1fields
      initEgg2Fields
      showEgg1Page2of2
    else
      initegg1fields
      initEgg2Fields
      showEgg1Page1of2
    end if
    cursor -1
    exit
  end if
end
```

```
--on mouseDown
--  global gNextStart
--  if legalButtonHandler() then
--    cursor 4
--    if gNextStart > 0 then
--      initegg1fields
--      initEgg2Fields
--      putUpMoreegg1Data
--      set gNextStart = 0
--    else
--      initegg1fields
--      initEgg2Fields
--      displayeggdata
--    end if
--    cursor -1
--  end if
--end
```

Score Script94

```
on exitFrame  
  cursor -1  
  checkDataPrintButton  
  go to the frame  
end
```

Score Script95

```

on initEggTask1Fields -- empty EggTask1 data fields of all but first Line
global gEggTask1Fields, gGamesViewedList, gTheFont
if not getOne(gGamesViewedList, #egg1) then
    append(gGamesViewedList, #egg1)
    repeat with thisfield in gEggTask1Fields
        put " " into field thisfield
        set the fontStyle of field thisfield = "plain"
        set NumLines = the number of lines of field thisfield
        set the font of field thisfield = gTheFont
        set the fontsize of field thisfield = 12

        case (thisfield) of
            "DataDate":Put "Date:" into text
            "EggTask1":Put "Task (1):" into text
            "EggVowels":Put "Vowels" into text
            "Egg1DiffScore":Put Quote &"Different"&Quote && "Cuml. Score:" into text
            "Egg1SameScore":Put Quote &"Same"&Quote && "Cuml. Score:" into text
        end case
        put text&Return into line 1 of field thisfield
        set the fontStyle of line 1 of field thisfield = "underline"
        delete line 2 to numlines of field thisfield
        put " " INTO LINE 2 OF field thisfield
        setPlainStyle(2,thisfield)
    end repeat
else
    repeat with thisfield in gEggTask1Fields
        set the fontStyle of line 1 of field thisfield = "UnderLine"
        if Line 2 of field thisfield = " " then next repeat
        set NumLines = the number of lines of field thisfield
        delete line 2 to numlines of field thisfield
        put " " INTO LINE 2 OF field thisfield
        setPlainStyle(2,thisfield)
    end repeat
end if

end

```

Movie Script100:CoalFieldInit

```
on initCoal1Fields
  global gCoal1Fields, gGamesViewedList, gTheFont
  if not getOne(gGamesViewedList, #Coal1) then
    append gGamesViewedList, #Coal1
    repeat with thisfield in gCoal1Fields
      put " " into field thisfield
      set NumLines = the number of lines of field thisfield
      set the font of field thisfield = gTheFont
      set the fontsize of field thisfield = 12
      -- set the fontStyle of line 1 of field thisfield = "underline"
      case (thisfield) of
        "DataDate":Put "Date:" into text
        "CoalTask1":Put "Task (1):" into text
          set the rect of member thisfield = rect(0, 0, 412, 336)
        "CoalContext":Put "Context:" into text
          set the rect of member thisfield = rect(0, 0, 473, 336)
        "CoalScore":Put "Cuml. Score:" into text
          set the rect of member thisfield = rect(0, 0, 95, 336)
      end case
      put text into line 1 of field thisfield
      set the fontStyle of line 1 of field thisfield = "underline"
      delete line 2 to numlines of field thisfield
      put " " INTO LINE 2 OF field thisfield
      set the fontstyle of line 2 of field thisfield to "plain"
    end repeat
  else
    repeat with thisfield in gCoal1Fields
      set NumLines = the number of lines of field thisfield
      if line 2 of field thisfield = " " then next repeat
      delete line 2 to numlines of field thisfield
      put " " INTO LINE 2 OF field thisfield
      set the fontstyle of line 2 of field thisfield to "plain"
    end repeat
  end if
end
```

```
on initCoal2Fields
  global gCoal2Fields, gGamesViewedList, gTheFont
  if not getOne(gGamesViewedList, #Coal2) then
    append gGamesViewedList, #Coal2
    repeat with thisfield in gCoal2Fields
      put " " into field thisfield
      set the font of field thisfield = gTheFont
      set the fontsize of field thisfield = 12
      set the fontStyle of line 1 of field thisfield = "underline"
      case (thisfield) of
        "DataDate":Put "Date:" into text
        "CoalTask2":Put "Task (2):" into text
        "Coal2TargPho":Put "Target Phoneme:" into text
        "Coal2Score":Put "Cuml. Scores:" into text
      end case
      put text into line 1 of field thisfield
      set the fontStyle of line 1 of field thisfield = "underline"
      put " " INTO LINE 2 OF field thisfield
      setplainstyle(2,thisfield)
    end repeat
  else
    repeat with thisfield in gCoal2Fields
```

```
set NumLines = the number of lines of field thisField
if line 2 of field thisField = " " then next repeat
delete line 2 to numlines of field thisfield
put " " INTO LINE 2 OF field thisField
setplainstyle(2,thisField)
end repeat
end if
end
```

Script of Cast Member102:EggBasket-2 SOS.

Movie Script104

```
on checkCCField
-- repeat with x = 1 to 56
--   put x && word 2 of item 1 of line x of field "CCLevelKeywords"
-- end repeat
repeat with x = 57 to 74
  put x && item 2 of line x of field "CCLevelKeywords"
end repeat
end
```

Movie Script107:setFieldRectScript

```
on setFieldRects
  -- hardwire all field rects so that they dont
  -- lose their sizing
  -- calls other handlers per screen
  set the rect of member "specifics" = rect(0,0,506,80)
  set the rect of member "specificsTitles" = rect(0, 0, 200, 100)
  -- set the rect of member "nextBut" = rect(0, 0, 98, 13)
  -- set the rect of member "prevBut" = rect(0, 0, 107, 13)
  -- set the rect of member "ShowMoreButton" = rect(0, 0, 107, 13)
  setEggRects
  setRapTapRects
  setCoalRects
  setCatConRects
  setBalloonRects
  setRTrects
end

on setEggRects
  set the rect of member "EggBasket-1 SOS" = rect(0,0,608,70)
  set the rect of member "DataDate" = rect(0, 0, 539, 32)
  set the rect of member "EggTask1" = rect(0, 0, 473, 304)
  set the rect of member "EggVowels" = rect(0, 0, 52, 304)
  set the rect of member "Egg1DiffScore" = rect(0, 0, 141, 304)
  set the rect of member "Egg1SameScore" = rect(0, 0, 129, 304)
  set the rect of member "EggBasket-2 SOS" = rect(0, 0, 633, 96)
  set the rect of member "DataDate" = rect(0, 0, 539, 32)
  set the rect of member "EggTask2" = rect(0, 0, 473, 272)
  set the rect of member "EggDuration" = rect(0, 0, 68, 272)
  set the rect of member "EggAmplification" = rect(0, 0, 112, 272)
  set the rect of member "EggSteps" = rect(0, 0, 88, 272)
  set the rect of member "Egg2DiffScore" = rect(0, 0, 121, 272)
  set the rect of member "Egg2SameScore" = rect(0, 0, 132, 272)
end

on setRapTapRects
  set the rect of member "RapTap1 SOS" = rect (0,0,608,56)
  set the rect of member "RapTap2 SOS" = rect (0,0,629,42)
  set the rect of member "DataDate" = rect(0, 0, 539, 32)
  set the rect of member "RapTapTask1" = rect(0, 0, 473, 224)
  set the rect of member "RapTap1Units" = rect(0, 0, 473, 224)
  set the rect of member "RapTap1Stimulus" = rect(0, 0, 473, 224)
  set the rect of member "RapTap1Interval" = rect(0, 0, 473, 224)
  set the rect of member "RapTap1FeedBack" = rect(0, 0, 83, 224)
  set the rect of member "RapTap1Score" = rect(0, 0, 75, 224)
  set the rect of member "RapTapTask2" = rect(0, 0, 473, 96)
  set the rect of member "RapTap2Units" = rect(0, 0, 473, 96)
  set the rect of member "RapTap2Stimulus" = rect(0, 0, 473, 96)
  set the rect of member "RapTap2FeedBack" = rect(0, 0, 124, 96)
  set the rect of member "RapTap2Score" = rect(0, 0, 75, 96)
end

on setCoalRects
  set the rect of member "CoalCarl SOS" = rect(0, 0, 554, 48)
  set the rect of member "Coal2 SOS" = rect(0, 0, 554, 48)
  set the rect of member "DataDate" = rect(0, 0, 539, 32)
  set the rect of member "CoalTask1" = rect(0, 0, 412, 336)
  set the rect of member "CoalContext" = rect(0, 0, 473, 336)
```



```

set the rect of member "Coal1Score" = rect(0, 0, 95, 336)
set the rect of member "CoalTask2" = rect(0, 0, 412, 320)
set the rect of member "Coal2TargPho" = rect(0, 0, 113, 320)
set the rect of member "Coal2Score" = rect(0, 0, 87, 320)
end

on setCatConRects
set the rect of member "Caterpillar Connection SOS" = rect(0, 0, 554, 32)
set the rect of member "DataDate" = rect(0, 0, 539, 32)
set the rect of member "CatConTask" = rect(0, 0, 72, 336)
set the rect of member "CatConUnits" = rect(0, 0, 81, 336)
set the rect of member "CatConInterval" = rect(0, 0, 58, 336)
set the rect of member "CatConTarget" = rect(0, 0, 104, 336)
set the rect of member "CatConFoils" = rect(0, 0, 58, 336)
set the rect of member "CatConScore" = rect(0, 0, 77, 336)
end

on setBalloonRects
set the rect of member "Balloons SOS" = rect(0, 0, 608, 48)
set the rect of member "DataDate" = rect(0, 0, 539, 32)
set the rect of member "BalloonTask" = rect(0, 0, 285, 304)
set the rect of member "BalloonNumber" = rect(0, 0, 58, 304)
set the rect of member "BalloonStimType" = rect(0, 0, 189, 304)
set the rect of member "BalloonVisDisplay" = rect(0, 0, 165, 304)
set the rect of member "BalloonNoise" = rect(0, 0, 285, 304)
set the rect of member "BalloonScore" = rect(0, 0, 77, 304)
end

on setRTrects
set the rect of member "Rhyme Time SOS2" = rect(0, 0, 605, 48)
set the rect of member "Rhyme Time SOS1" = rect(0, 0, 630, 32)
set the rect of member "DataDate" = rect(0, 0, 539, 32)
set the rect of member "RT Task(1)" = rect(0, 0, 285, 112)
set the rect of member "RT InSetOf" = rect(0, 0, 83, 112)
set the rect of member "RT BgNoise1" = rect(0, 0, 116, 112)
set the rect of member "RT Score1" = rect(0, 0, 85, 112)
set the rect of member "RT Task(2)" = rect(0, 0, 337, 128)
set the rect of member "RT responseChoice" = rect(0, 0, 120, 128)
set the rect of member "RT BgNoise2" = rect(0, 0, 116, 128)
set the rect of member "RT Score2" = rect(0, 0, 85, 128)
end

```

Script of Cast Member108

Movie Script109

```

on checkEggKeyField
repeat with x = 1 to 30
put x && item 2 of line x of field "EggLevelKeywords"
end repeat
end

```

Movie Script110

```
on replace
  repeat with x = 58 to 86
    put "/ra-la/" into word 2 of item 1 of line x of field "egglevelKeywords"
  end repeat
  repeat with x = 87 to 115
    put "/ma-na/" into word 2 of item 1 of line x of field "egglevelKeywords"
  end repeat
end
```

Score Script113

```
on exitFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  puppetButtonSprites true
  spritesonlist [28]
end
```

Score Script114

```
on exitFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  puppetBut onSprites true
  cursor 4
  initEgg1Fields
  initEgg2Fields
  displayEggData
end
```

Movie Script117

```
on initEggFields -- empty EggBasket's data fields of all but first Line
global gEggTask1Fields
repeat with thisfield in gEggTask1Fields
  set NumLines = the number of lines of field thisfield
  set the font of field thisfield = "Helvetica"
  set the fontsize of field thisfield = 12
  set the fontStyle of line 1 of field thisfield = "underline"
  case (thisfield) of
    "DataDate":Put "Date:" into text
    "EggTask1":Put "Task :" into text
    "EggVowels":Put "Number:" into text
    "Egg1DiffScore":Put "Stimulous Type:" into text
    "Egg1SameScore":Put "Visual Display:" into text
  end case
  put text into line 1 of field thisfield
  delete line 2 to numlines of field thisfield
  put " " INTO LINE 2 OF field thisfield

  set the fontstyle of line 2 of field thisfield to "plain"
end repeat
end
```

Movie Script118

```
on initEgg2Fields start -- empty EggTask2 data fields of all but first Line
global gEggTask2Fields, gGamesViewedList, gTheFont
if not getOne(gGamesViewedList, #Egg2) then
  append gGamesViewedList, #Egg2
  repeat with thisfield in gEggTask2Fields
    put " " into field thisfield
    set the fontStyle of line 1 of field thisfield = "Plain"
    set the font of field thisfield = gTheFont
    set the fontsize of field thisfield = 12

    case (thisField) of
      "DataDate":Put "Date:" into text
      "EggTask2":Put "Task (2):" into text
      "EggDuration":Put "Duration:" into text
      "EggAmplification":Put "Amplification:" into text
      "EggSteps":Put "Steps:" into text
      "Egg2DiffScore":Put Quote &"Diff."&Quote && "Cuml. Score:" into text
      "Egg2SameScore":Put Quote &"Same"&Quote && "Cuml. Score:" into text
    end case
    put text&return into line 1 of field thisField
    set the fontStyle of line 1 of field thisfield = "underline"
    put " " INTO LINE 2 OF field thisField
    setPlainStyle(2, ThisField)
  end repeat
else
  repeat with thisfield in gEggTask2Fields
    set NumLines = the number of lines of field thisField
    if line 2 of field thisField = " " then next repeat
    delete line 2 to numlines of field thisfield
    put " " INTO LINE 2 OF field thisField
    setPlainStyle(2, ThisField)
  end repeat
end if
end
```

Score Script119

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    set gSessionNum = gSessionNum - 1
    go to frame "Basket Full of Eggs Data"
  end if
end
```

Score Script121

Movie Script122:DisplayRapTapData

```
on DisplayRapTapData
  global gRecordDisplay, gWhichUser, gSessionNum, gTask2Sprites, gRapTapKeyWords
  if VoidP(gRapTapKeyWords) then put field "RapTapLevelKeyWords" into gRapTapKeyWords

  put getData (gRecordDisplay, gWhichUser, 4) into dataList

  set SessionNums = count(dataList)

  putUpNavArrows SessionNums

  MakeVisRapTapSprites

  updatestage
  put getpropat(dataList, gSessionNum) into date
  put getProp(dataList,date) into sessionList
  put date into line 2 of field "dataDate"
  setPlainStyle(2,"dataDate")
  set NumRounds = count(sessionList)
  set Task1 = 0
  set task2 = 0
  repeat with x = 1 to numRounds
    put getpropat (sessionList,x) into Level
    set LineNum1 = the number of lines of field "RapTap1Score"
    set LineNum2 = the number of lines of field "RapTap2Score"
    case (true) of
      (value(Level)) <= (12): PutUpRapTask1Stats level
        put getProp (sessionList,level) & return into line linenum1 of field
        "RapTap1Score"
        setPlainStyle(linenum1,"RapTap1Score")
        set task1 = 1
      (value(Level)) >= (13): PutUpRapTask2Stats level
        put getProp (sessionList,level) & return into line linenum2 of field
        "RapTap2Score"
        setPlainStyle(linenum2,"RapTap2Score")
        set task2 = 1
    end case
  end repeat

  if (task1 = 1) and (task2 = 1) then -- both tasks must be displayed
    if numrounds = 16 then -- full screen so save space!!
      if the machinetype = 256 then
        set NewTop = (the bottom of sprite 11) - 15
      else
        set NewTop = (the bottom of sprite 11) - 8
        -- mac needs less space
      end if
    else
      set NewTop = (the bottom of sprite 11) - 8
    end if
    set the loc of sprite 30 = point(69,newTop)
    set the loc of sprite 31 = point(155,newTop)
    set the loc of sprite 32 = point(270,newTop)
    set the loc of sprite 33 = point(400,newTop)
    set the loc of sprite 34 = point(553,newTop)
  end if

  if Task1 = 1 and task2 = 0 then -- just display task 1
    repeat with x in gTask2Sprites
```

```

    set the loc of sprite x = point(-1000,-1000)
  end repeat
  updatetage
  repeat with x = 11 to 16
    set the Visible of sprite x = true
  end repeat
  updatetage
  exit
end if
if task1 = 0 and task2 = 1 then -- just display task 2
  repeat with x = 11 to 16
    set the Visible of sprite x = false
  end repeat
  updatetage
  set NewTop = 168
  set the loc of sprite 30 = point(69,newTop)
  set the loc of sprite 31 = point(155,newTop)
  set the loc of sprite 32 = point(270,newTop)
  set the loc of sprite 33 = point(400,newTop)
  set the loc of sprite 34 = point(553,newTop)
end if
end

```

```

on PutUpRapTask1Stats whichLine
  global gRapTapKeyWords
  set whichLine = Value(whichLine)

```

```

  set lineNum = the number of lines of field "RapTapTask1"
  put item 1 of line whichLine of gRapTapKeyWords & return into line lineNum of field
  "RapTapTask1"
  setPlainStyle(lineNum, "RapTapTask1")
  set lineNum = the number of lines of field "RapTap1Units"
  put item 2 of line whichLine of gRapTapKeyWords & return into line lineNum of field
  "RapTap1Units"
  setPlainStyle(lineNum, "RapTap1Units")
  set lineNum = the number of lines of field "RapTap1Stimulus"
  put item 3 of line whichLine of gRapTapKeyWords & return into line lineNum of field
  "RapTap1Stimulus"
  setPlainStyle(lineNum, "RapTap1Stimulus")
  set lineNum = the number of lines of field "RapTap1Interval"
  put item 4 of line whichLine of gRapTapKeyWords & return into line lineNum of field
  "RapTap1Interval"
  setPlainStyle(lineNum, "RapTap1Interval")
  set lineNum = the number of lines of field "RapTap1FeedBack"
  put item 5 of line whichLine of gRapTapKeyWords & return into line lineNum of field
  "RapTap1FeedBack"
  setPlainStyle(lineNum, "RapTap1FeedBack")
end

```

```

on PutUpRapTask2Stats whichLine
  global gRapTapKeyWords
  set whichLine = Value(whichLine)

```

```

  set lineNum = the number of lines of field "RapTapTask2"
  put item 1 of line whichLine of gRapTapKeyWords & return into line lineNum of field
  "RapTapTask2"
  setPlainStyle(lineNum, "RapTapTask2")

```



```

    set lineNum = the number of lines of field "RapTap2Units"
    put item 2 of line whichLine of gRapTapKeyWords & return into line lineNum of field
    "RapTap2Units"
    setPlainStyle(lineNum, "RapTap2Units")
    set lineNum = the number of lines of field "RapTap2Stimulus"
    put item 3 of line whichLine of gRapTapKeyWords & return into line lineNum of field
    "RapTap2Stimulus"
    setPlainStyle(lineNum, "RapTap2Stimulus")
    set lineNum = the number of lines of field "RapTap2FeedBack"
    put item 4 of line whichLine of gRapTapKeyWords & return into line lineNum of field
    "RapTap2FeedBack"
    setPlainStyle(lineNum, "RapTap2FeedBack")
end

```

Score Script130

```

on exitFrame
    MakeVisRapTapSprites -- special case from RapTap task 2
    global gTask2Sprites
    set gTask2Sprites = {30,31,32,33,34}
    spritesonList gTask2Sprites
    puppetButtonSprites true
    -- initRaptapFields
end

```

Score Script131

```

on exitFrame
    cursor 4
    initRaptapFields
    displayRapTapData
    cursor -1
end

```

Score Script132

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    spriteListOff
    updateStage
    -- initRaptapFields
    set gSessionNum = gSessionNum + 1
    go to Marker (0)
  end if
end
```

Score Script133

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    spriteListOff
    updateStage
    -- initRaptapFields
    set gSessionNum = gSessionNum - 1
    go to Marker (0)
  end if
end
```

Movie Script134:setPlainStyleScript

```
on setPlainStyle whichLine, whichField
  --use after setting font style of a line of a field
  -- on the PC once a style is set it sticks unless you
  -- use this script on every line thereafter
  set the fontStyle of Line whichLine of field whichField = "Plain"
end
```

```

on initRapTapFields -- empty RapTaps data fields of all but first Line
  global gRapTapFields, gGamesViewedList, gTheFont
  set fieldCount = count(gRapTapFields)
  if not getOne(gGamesViewedList, #RapTap) then
    append gGamesViewedList, #rapTap

    repeat with x = 1 to fieldCount
      set thisfield = getPropat(gRapTapFields, x)
      set Y = (the number of lines of field thisfield)
      put " " into field thisfield
      set the font of field thisfield = gTheFont
      set the fontsize of field thisfield = 12
      set the fontStyle of Line 1 of field thisfield = "underline"
      set text = getaprop(gRapTapFields, thisfield)
      put text & return into field thisfield
      set the fontStyle of Line 1 of field thisfield = "underline"
      delete Line 2 to Y of field thisfield
      put " " into line 2 of field thisfield
      setPlainStyle(2, thisfield)
    end repeat
  else
    repeat with x = 1 to fieldCount
      set thisfield = getPropat(gRapTapFields, x)
      set Y = (the number of lines of field thisfield)
      if Line 2 of field thisfield = " " then next repeat
      delete Line 2 to Y of field thisfield
      put " " into line 2 of field thisfield
      setPlainStyle(2, thisfield)
    end repeat
  end if
end

```

Score Script138

```
on exitFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  cursor 4
  initCoallFields
  initCoal2Fields
  displayCoalCarData
end
```

Score Script139

```
on exitFrame
  MakeVisRapTapSprites -- special case from RapTap task 2
  puppetButtonSprites true
  spritesonlist [28]
end
```

Score Script140

```
on exitFrame
  cursor 4
  PutUpCoallData
  cursor -1
end
```

Score Script141

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    spriteListOff
    updatestage
    set gSessionNum = gSessionNum + 1
    go to frame "C.C. Coal Car Data"
  end if
end
```

Score Script142

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    spriteListOff
    updateStage
    set gSessionNum = gSessionNum - 1
    go to Frame "C.C. Coal Car Data"
  end if
end
```

Score Script152

```
on mouseDown
  global gPageNum, gNumPages
  if legalButtonHandler() then
    cursor 4
    if gPageNum = 1 then
      if gNumPages = 2 then
        initcoallfields
        showCCPage2of2
      else
        initcoallfields
        showCCPage2of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 2 then
      if gNumPages = 2 then
        initcoallfields
        showCCPage1of2
      else
        initcoallfields
        showCCPage3of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 3 then
      initcoallfields
      showCCPage1of3
      cursor -1
      exit
    end if

  end if
end
```

```
--on mouseDown
-- global gNextStart
-- if legalButtonHandler() then
--   cursor 4
--   if gNextStart > 0 then
--     initcoallfields
--
--     putUpMorecoallData
--     set gNextStart = 0
--   else
--     initcoallfields
--
--     PutUpcoallData
--   end if
--   cursor -1
-- end if
--end
```

Movie Script154

```
on DisplayCoalCarData
  global gRecordDisplay, gWhichUser, gSessionNum, gNextStart, gMoreButton, gMoreButtonloc
  put getData (gRecordDisplay, gWhichUser, 6) into dataList
  -- put dataList
  -- set dataList = convertList (dataList, 57) ---!!!
  -- put dataList
  set SessionNums = count(dataList)
  --- putUpNavArrows SessionNums

  updatestage

  put getpropat(dataList, gSessionNum) into date
  put getat(dataList, gSessionNum) into SessionList
  -- here we need to see if the scores for this session are in the
  -- first task or the second. Levels <= 56 are task 1, Levels > 56 are task 2
  set levelList = []
  set numRounds = count(sessionList)
  repeat with x = 1 to numRounds
    append levelList, getPropat(sessionList, x)
  end repeat
  if max(levelList) < 57 then -- task 1
    go to frame "coalCar1"
  end if
  if Min(levelList) > 56 then -- task 2
    go to frame "coalCar2"
  end if
end
```

Score Script155

```
on exitFrame  
  cursor 4  
  PutUpCoal2Data  
  cursor -1  
end
```


Movie Script156

```
on PutUpCoal2Data
  global gRecordDisplay, gWhichUser, gSessionNum, gNextStart, gMoreButton, gMoreButtonloc

  put getData (gRecordDisplay, gWhichUser, 6) into dataList
  set SessionNums = count(dataList)

  putUpNavArrows SessionNums

  updatestage

  put getpropat(dataList, gSessionNum) into date
  put getat(dataList, gSessionNum) into SessionList
  put date into line 2 of field "dataDate"
  setPlainStyle(2, "dataDate")
  set numRounds = count(SessionList)

  repeat with x = 1 to numRounds
    put getpropat (sessionList, x) into Level
    set LineNum = the number of lines of field "Coaltask2"
    PutUpCoal2words level
    put getProp (sessionList, level) & return into line linenum of field "Coal2score"
    setPlainStyle(lineNum, "Coal2score")
  end repeat
end

on PutUpCoal2words whichLine
  set WhichLine = value(whichLine)
  put field "CCKeywords" into keywords
  set lineNum = the number of lines of field "Coaltask2"
  put item 1 of line whichLine of keywords & return into line lineNum of field
  "Coaltask2"
  setPlainStyle(lineNum, "Coaltask2")
  set lineNum = the number of lines of field "Coal2TargPho"
  put item 2 of line whichLine of keywords & return into line lineNum of field
  "Coal2TargPho"
  setPlainStyle(lineNum, "Coal2TargPho")
end
```

Score Script157

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    spriteListOff
    updatestage
    set gSessionNum = gSessionNum + 1
    go to frame "C.C. Coal Car Data"
  end if
end
```

Score Script158

```
on mouseDown
  global gSessionNum
  if BlankArrow() then exit -- do nothing if blank arrow is up
  if legalButtonHandler() then
    spriteListOff
    updatestage
    set gSessionNum = gSessionNum - 1
    go to Frame "C.C. Coal Car Data"
  end if
end
```

Score Script159

```
on mouseDown
  global gPageNum, gNumPages

  if legalButtonHandler() then
    cursor 4
    if gPageNum = 1 then
      if gNumPages = 2 then
        initCatConfields
        showCatConPage2of2
      else
        initCatConfields
        showCatConPage2of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 2 then
      if gNumPages = 2 then
        initCatConfields
        showCatConPage1of2
      else
        initCatConfields
        showCatConPage3of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 3 then
      initCatConfields
      showCatConPage1of3
      cursor -1
      exit
    end if

  end if
end
```


Movie Script162:getRectsUtility

```
on getegg1Rects
  global gEggTask1Fields
  put " " into field "tempRects"

  repeat with thisfield in gEggTask1Fields
    put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
    the rect of member thisfield & return after field "tempRects"
  end repeat
end

on getegg2Rects
  global gEggTask2Fields
  put " " into field "tempRects"

  repeat with thisfield in gEggTask2Fields
    put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
    the rect of member thisfield & return after field "tempRects"
  end repeat
end

on getCoallRects
  global gCoallFields
  put " " into field "tempRects"

  repeat with thisfield in gCoallFields
    put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
    the rect of member thisfield & return after field "tempRects"
  end repeat
end

on getCoal2Rects
  global gCoal2Fields
  put " " into field "tempRects"

  repeat with thisfield in gCoal2Fields
    put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
    the rect of member thisfield & return after field "tempRects"
  end repeat
end

on getCatConRects
  global gCatConFields
  put " " into field "tempRects"

  repeat with thisfield in gCatConFields
    put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
    the rect of member thisfield & return after field "tempRects"
  end repeat
end

on getBalloonRects
  global gBalloonfields
```

```

put " " into field "tempRects"

repeat with thisfield in gBalloonfields
    put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
the rect of member thisfield & return after field "tempRects"
end repeat
end

on getRTRects
    global gRTFieldList
    put " " into field "tempRects"

    repeat with thisfield in gRTFieldList
        put "set the rect of member"&&Quote&thisfield&Quote&&"="&& ~
the rect of member thisfield & return after field "tempRects"
    end repeat
end

```

Score Script174

```

on mouseDown
    global gSessionNum
    if BlankArrow() then exit -- do nothing if blank arrow is up
    if legalButtonHandler() then
        set gSessionNum = gSessionNum + 1
        go to frame "Basket Full of Eggs Data"
    end if
end

```

Score Script175

```

on mouseDown
    global gSessionNum
    if BlankArrow() then exit -- do nothing if blank arrow is up
    if legalButtonHandler() then
        set gSessionNum = gSessionNum + 1
        go to frame "Basket Full of Eggs Data"
    end if
end

```

Score Script176

Score Script178

```
on mouseDown
  global gPageNum, gNumPages
  if legalButtonHandler() then
    cursor 4
    if gPageNum = 1 then
      if gNumPages = 2 then
        initegg1fields
        initEgg2Fields
        showEgg2Page2of2
      else
        initegg1fields
        initEgg2Fields
        showEgg2Page2of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 2 then
      if gNumPages = 2 then
        initegg1fields
        initEgg2Fields
        showEgg2Page1of2
      else
        initegg1fields
        initEgg2Fields
        showEgg2Page3of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 3 then
      initegg1fields
      initEgg2Fields
      showEgg2Page1of3
      cursor -1
      exit
    end if

  end if
end
```

```
--on mouseDown
-- global gNextStart
-- if legalButtonHandler() then
--   cursor 4
--   if gNextStart > 0 then
--     initegg1fields
--     initEgg2Fields
--     putUpMoreegg2Data
--     set gNextStart = 0
--   else
--     initegg1fields
--     initEgg2Fields
--     displayeggdata
--   end if
--   cursor -1
```

```
-- end if  
--end
```

Score Script179

Score Script180

```
on mouseDown
  global gPageNum, gNumPages
  if legalButtonHandler() then
    cursor 4
    if gPageNum = 1 then
      if gNumPages = 2 then
        initBalloonfields
        showBalloonPage2of2
      else
        initBalloonfields
        showBalloonPage2of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 2 then
      if gNumPages = 2 then
        initBalloonfields
        showBalloonPage1of2
      else
        initBalloonfields
        showBalloonPage3of3
      end if
      cursor -1
      exit
    end if

    if gPageNum = 3 then
      initBalloonfields
      showBalloonPage1of3
      cursor -1
      exit
    end if

  end if
end
end
```

```
--on mouseDown
-- global gNextStart
-- if legalbuttonhandler() then
--   cursor 4
--   if gNextStart > 0 then
--     initBalloonFields
--     putUpMoreBalloonData
--     set gNextStart = 0
--   else
--     initBalloonFields
--     displayBalloondata
--   end if
--   cursor -1
-- end if
--end
```

Score Script181

```
on mouseUp  
end
```

Movie Script182

```
on putUpPg1of2  
    put "Pg.1 of 2" into line 3 of field "dataDate"  
    setPlainStyle(3,"dataDate")  
end  
  
on putUpPg2of2  
    put "Pg.2 of 2" into line 3 of field "dataDate"  
    setPlainStyle(3,"dataDate")  
end  
  
on putUpPg1of3  
    put "Pg.1 of 3" into line 3 of field "dataDate"  
    setPlainStyle(3,"dataDate")  
end  
  
on putUpPg2of3  
    put "Pg.2 of 3" into line 3 of field "dataDate"  
    setPlainStyle(3,"dataDate")  
end  
  
on putUpPg3of3  
    put "Pg.3 of 3" into line 3 of field "dataDate"  
    setPlainStyle(3,"dataDate")  
end
```

Score Script184

```
on mouseUp  
end
```

Parent Script185:Egg2FieldMemberNums

```
property dataDate,EggTask2,EggDuration,EggAmplification,EggSteps,  
property Egg2DiffScore,Egg2SameScore
```

```
-- purpose is to return the member number of each on screen field while  
-- building the data on screen. In hopes of speeding up screen display
```

```
on new Me  
  initProps me  
  return me
```

```
end
```

```
on initProps me  
  global gEggTask2Fields  
  repeat with thisField in gEggTask2Fields  
    case (thisField) of  
      "DataDate": put the membernum of member "DataDate" into datadate  
      "EggTask2": put the membernum of member "EggTask2" into EggTask2  
      "EggDuration": put the membernum of member "EggDuration" into EggDuration  
      "EggAmplification": put the membernum of member "EggAmplification" into  
EggAmplification  
      "EggSteps": put the membernum of member "EggSteps" into EggSteps  
      "Egg2DiffScore": put the membernum of member "Egg2DiffScore" into Egg2DiffScore  
      "Egg2SameScore": put the membernum of member "Egg2SameScore" into Egg2SameScore  
      otherwise: alert "there may be a problem with EGG 2 fields. Check gEggTask2Fields-  
for changed field names"  
    end case  
  end repeat  
end
```

```
on FieldNum me, whichField  
  case whichField of  
    #DataDate: return datadate  
    #EggTask2: return EggTask2  
    #EggDuration: return EggDuration  
    #EggAmplification: return EggAmplification  
    #EggSteps: return EggSteps  
    #Egg2DiffScore: return Egg2DiffScore  
    #Egg2SameScore: return Egg2SameScore  
  end case  
end
```

```
on testobj howmanyTimes  
  global gEgg2ob  
  put the ticks into start  
  repeat with x = 1 to howmanytimes  
    set b = the number of lines of field "eggamplification"  
  end repeat  
  put the ticks - start  
  put the ticks into start  
  repeat with x = 1 to howmanytimes  
    set b = the number of lines of field fieldnum(gegg2ob, #eggamplification)  
  end repeat  
  put the ticks - start  
end repeat
```

```

on printDataPage
    -- Main handler to create printout of one screen's data
    -- Checks frame label of current frame to determin which sprites
    -- need to be printed and how to align each field.
    -- important that the label names not be changed!!

    cursor 4
    set doc = 0
    set doc = new(xtra "printomatic")

    if not objectP(doc) then
        alert "there is a problem with printing"
        cursor -1
        exit
    end if
    put the font of member "specifics" into theFont
    reset doc
    setLandscapeMode (doc,true)
    setMargins(doc,rect(72,36,72,36))
    newPage doc

    setTextfont doc, theFont

    set printingList = GetPrintingList() -- fine this function below
    -- returns property list of the fields to print
    -- the property is a symbol #L,#C or #R for alignment of left, centered etc.
    set fieldCount = count(printingList)
    repeat with x = 1 to fieldCount
        set fieldSprite = getat(printingList,x)
        if the visible of sprite fieldSprite = false then next repeat -- special case, RapTap
    task 2
        set alignHow = getpropat(printingList,x)
        case alignHow of
            #L: setTextJust doc, "left"
            #R: setTextJust doc, "Right"
            #C: setTextJust doc, "centered"
            #S: DoSpritePrint fieldSprite
        end case
        -- set memberName = the Name of The Member of sprite fieldSprite
        set spriteRect = the rect of sprite fieldSprite
        newFrame Doc, spriteRect, false
        append doc, sprite fieldSprite, false
    end repeat
    -- now put in criteria at bottom of page.
    set CriteRect = rect(11, 465, 631, 540)
    setTextJust doc, "Centered"
    newFrame Doc, CriteRect, false
    set CriteText = getCriteText()
    -- returns name of cast member with
    -- text for that page's criteria
    append doc, member CriteText, false
    setDocumentname (doc, "Earobics PRO data") -- change here to put different name
    -- on print progress dialogues.
    print doc
    set doc = 0
    cursor -1
end

```



```

on DoSpritePrint whichSprite
  global doc
  set spriteRect = the rect of sprite whichSprite
  newFrame Doc, spriteRect, false
  append doc, sprite whichSprite
end

on getPrintingList
  -- set these lists up by hand. Property is alignment of
  -- field sprite, Value is the sprite num of field sprite
  -- Make sure label names are correct
  set printingList = [:]
  case (label(0)) of
    (label("CoalCarl")): set PrintingList = [#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #L:10, #C:11]
    (label("CoalCar2")): set PrintingList = [#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #C:10, #C:11]
    (label("Rap-a-Tap-Tap Data")): set PrintingList =
[#L:4, #L:5, #L:6, #L:7, #L:8, #L:10, #L:11, #L:12, #L:13, #L:14, #C:15, #C:16, #L:30, #L:31, #L:32, #C:
33, #C:34]
    (label("Caterpillar Connection Data")): set PrintingList =
[#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #L:10, #C:11, #C:12, #C:13, #C:14]
    (label("Karloon's Balloons Data")): set PrintingList =
[#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #C:10, #L:11, #L:12, #L:13, #C:14]
    (label("eggTask1")): set PrintingList =
[#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #C:10, #C:11, #C:12]
    (label("eggTask2")): set PrintingList =
[#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #L:10, #L:11, #C:12, #C:13, #C:14]
    (label("Rhyme Time Data")): set PrintingList
= [#L:4, #L:5, #L:6, #L:7, #L:8, #L:9, #L:10, #L:11, #C:12, #
#C:13, #L:15, #L:16, #C:17, #C:18]
  end case
  return PrintingList
end

```

```

on getCriteText
  -- same idea as above, but returns name of cast member with criterion text
  -- make sure member is there, and named correctly!!
  set text = 0
  case (Label(0)) of
    (label("eggTask1")): set text = "egg1Criteria"
    (label("eggTask2")): set text = "egg2Criteria"
    (label("Caterpillar Connection Data")): set text = "CatConCriteria"
    (label("Karloon's Balloons Data")): set text = "BalloonCriteria"
    (label("Rhyme Time Data")): set text = "RhymeTimeCriteria"
    (label("Rap-a-Tap-Tap Data")): set text = "RapTapCriteria"
    (label("CoalCarl")): set text = "CoalCarlCriteria"
    (label("CoalCar2")): set text = "CoalCar2Criteria"
  end case
  return text
end

```

```

on findSprite whichMemberName
  -- returns first sprite number containing "WhichMemberName"

```

```
-- or 0 if no such sprite.  
set theSprite = 0  
repeat with x = 1 to 48  
  if the member of sprite x = member whichMemberName then  
    set theSprite = x  
  exit repeat  
end if  
end repeat  
return theSprite  
end
```

Parent Script196:BalloonFieldMemberNums

```
property dataDate,BalloonTask,BalloonNumber,BalloonStimType,BalloonVisDisplay,  
property BalloonNoise,BalloonScore
```

```
-- purpose is to return the member number of each on screen field while  
-- building the data on screen. In hopes of speeding up screen display
```

```
--set gBalloonfields = ["DataDate", "BalloonTask", "BalloonNumber", "BalloonStimType",  
"BalloonVisDisplay", "BalloonNoise", "BalloonScore"]
```

```
on new Me  
  initProps me  
  return me
```

```
end
```

```
on initProps me  
  global gBalloonfields  
  repeat with thisField in gBalloonfields  
    case (thisField) of  
      "DataDate": put the membernum of member "DataDate" into datadate  
      "BalloonTask": put the membernum of member "BalloonTask" into BalloonTask  
      "BalloonNumber": put the membernum of member "BalloonNumber" into BalloonNumber  
      "BalloonStimType": put the membernum of member "BalloonStimType" into  
BalloonStimType  
      "BalloonVisDisplay": put the membernum of member "BalloonVisDisplay" into  
BalloonVisDisplay  
      "BalloonNoise": put the membernum of member "BalloonNoise" into BalloonNoise  
      "BalloonScore": put the membernum of member "BalloonScore" into BalloonScore  
      otherwise: alert "there may be a problem with EGG 2 fields. Check gEggTask2Fields~  
for changed field names"  
    end case  
  end repeat  
end
```

```
on FieldNum me, whichField  
  case whichField of  
    #DataDate: return datadate  
    #BalloonTask: return BalloonTask  
    #BalloonNumber: return BalloonNumber  
    #BalloonStimType: return BalloonStimType  
    #BalloonVisDisplay: return BalloonVisDisplay  
    #BalloonNoise: return BalloonNoise  
    #BalloonScore: return BalloonScore  
  end case  
end
```

Script of Cast Member197:prevBut

```
on mouseUp
    printDataPage
end
```

Score Script198

```
on mouseUp
    dontPassevent
end
```

Movie Script199

```
on MakeVisRapTapSprites
    -- script needs to be called every where
    -- since it is possible to leave some
    -- channels invisble when leaving
    -- RapTapScreen if only task 2 is up
    repeat with x = 11 to 16
        set the visible of sprite x = true
    end repeat
end
```

Score Script200

```
on exitFrame
    MakeVisRapTapSprites -- special case from RapTap task 2
    puppetButtonSprites true
    spritesonlist [28]
end
```

```

on putUpNavArrows sessionNums
-- evaluates which data session is on screen and puts
-- up the appropriate navigational arrows
global gSessionNum
-- first correct gSessionNum to pin out at first and last item
-- of sessionList by referencing parameter "sessionNums"
if gSessionNum >= SessionNums then
    set gSessionNum = SessionNums
end if
if gSessionNum < 1 then
    set gSessionNum = 1
end if

if gSessionNum = 1 then
    if sessionNums > 1 then
        putUpLeftArrow
    else
        putUpNoArrows -- only one or no records
    end if
else
    if gSessionNum > 1 and gSessionNum < sessionNums then
        putUpBothArrows
    else
        if gSessionNum = sessionNums then
            PutUpRightArrow
        end if
    end if
end if
end

```

```

on putUpBothArrows
global gPrevButton, gNextButton
set the member of sprite gNextButton to member "leftArrow"
set the member of sprite gPrevButton to member "RightArrow"
end

```

```

on PutUpLeftArrow
global gPrevButton, gNextButton
set the member of sprite gNextButton to member "leftArrow"
set the member of sprite gPrevButton to member "RightArrow Grey"
end

```

```

on PutUpRightArrow
global gPrevButton, gNextButton
set the member of sprite gNextButton to member "leftArrow Grey"
set the member of sprite gPrevButton to member "rightArrow"
end

```

```

on putUpNoArrows
global gPrevButton, gNextButton
set the member of sprite gPrevButton to member "RightArrow Grey"
set the member of sprite gNextButton to member "LeftArrow Grey"
end

```

```

on BlankArrow
  -- function called by Nav arrow sprite scripts
  -- to lock out button action if the Blank arrow is up
  put the name of the member of sprite the clickOn into temp
  if word 2 of temp = "grey" then
    return 1
  else
    return 0
  end if
end

on putUpViewPage2of2
  global gMoreButton, gMoreButtonLoc
  set the loc of sprite gMoreButton to gMoreButtonLoc
  set the member of sprite gMoreButton = member "View2of2Button"
  updatestage
end

on putUpViewPage1of2
  global gMoreButton, gMoreButtonLoc
  set the loc of sprite gMoreButton to gMoreButtonLoc
  set the member of sprite gMoreButton = member "View1of2Button"
  updatestage
end

on putUpViewPage1of3
  global gMoreButton, gMoreButtonLoc
  set the loc of sprite gMoreButton to gMoreButtonLoc
  set the member of sprite gMoreButton = member "View1of3Button"
  updatestage
end

on putUpViewPage2of3
  global gMoreButton, gMoreButtonLoc
  set the loc of sprite gMoreButton to gMoreButtonLoc
  set the member of sprite gMoreButton = member "View2of3Button"
  updatestage
end

on putUpViewPage3of3
  global gMoreButton, gMoreButtonLoc
  set the loc of sprite gMoreButton to gMoreButtonLoc
  set the member of sprite gMoreButton = member "View3of3Button"
  updatestage
end

on sendAwayViewPageButton
  global gMoreButton, gMoreButtonLoc
  set the loc of sprite gMoreButton to point(1000,1000)
  updatestage
end

```

Movie Script209:LegalButtonHandler

on LegalButtonHandler

```
set lRollover = TRUE
set clickSprite = the clickon
set lButtonUpName = the name of member (the member of sprite clickSprite)
set lButtonDownName = lButtonUpName & " Down"
set the member of sprite clickSprite = member lButtonDownName
updatestage

repeat while the stillDown = TRUE
  if rollover (clickSprite) = TRUE then
    set lRollover = TRUE
    set the membernum of sprite clickSprite = the number of member lButtonDownName
    updatestage
  else
    set lRollover = FALSE
    set the membernum of sprite clickSprite = the number of member lButtonupName
    updatestage
  end if
end repeat

if lRollover = FALSE then return 0

set the membernum of sprite clickSprite = the number of member lButtonupName
updatestage
return 1

end LegalButtonHandler
```

Script of Cast Member211:printButton

```
on mouseDown
  if legalButtonHandler() then
    openPrintPlace
  end if
end
```

Script of Cast Member213:UserButton

```
on mouseDown
  global MenuMaker, gWhichGame, gWhichUser, jimWhichGame, jimWhichUser
  set x = the clickOn
  -- set the ink of sprite x to 2
  updateStage
  set theLoc to point(the left of sprite x, the bottom of sprite x)
  set whichField = the membernum of member "userNameList"

  showMenu (gmenuMaker, whichField, theLoc)

  -- set the ink of sprite x to 0
  updateStage

  put the result into whichLine
  if whichLine = -1 or whichLine = #nothing then exit
  put line whichLine of field "userNameList" into gWhichUser
  delete char 1 of gWhichUser -- take out leading space
  if gWhichUser = jimWhichUser then exit
  put gWhichUser into jimWhichUser
  if jimWhichGame <> "" and jimWhichUser <> "" then
    spriteListOff
    goJim
  end if
end
```

Script of Cast Member214:GameButton

```
on mouseDown
  global MenuMaker, gWhichGame, gWhichUser, jimWhichGame, jimWhichUser
  set x = the clickOn
  -- set the ink of sprite x to 2
  updatestage
  set theLoc to point(the left of sprite x, the bottom of sprite x)
  set whichMember = the membernum of member "GameList"
  showMenu (gmenuMaker, whichMember, theLoc)
  -- set the ink of sprite x to 0
  updatestage

  put the result into whichLine

  if whichLine = -1 or whichLine = #nothing then exit
  put line whichLine of field "GameList" into gWhichGame
  delete char 1 of gWhichGame -- take out leading space
  if gWhichGame = jimWhichGame then exit
  put gWhichGame into jimWhichGame

  if jimWhichGame <> "" and jimWhichUser <> "" then
    goJim
  end if

end
```

Script of Cast Member215:help

```
on mouseDown
  global gNextButton, gPrevButton, gMoreButton, gDataPrintButton
  if legalButtonHandler() then
    puppetButtonSprites false
    puppetSprite gDataPrintButton, false
    set the loc of sprite gMoreButton = point (1000,1000)
    spriteListOff
    go to frame "menu"
  end if
end
```

Script of Cast Member247:MenuButton:

```
on mouseDown
  if legalButtonHandler() then
    spriteListOff
    go to movie "dataTest"
  end if
end
```

Script of Cast Member250:exit

```
on mouseDown
  Global gSpritesOnList
  if legalButtonHandler() then
    spriteListOff
    set gSpritesOnList = []
    cursor 4
    go to movie "dataTest"
  end if
end
```

Score Script4

```
on exitFrame  
  go to frame "Listen"  
end
```

Score Script5

```
on exitFrame  
  go to frame "Voweltest"  
end
```

Score Script6

```
on exitFrame  
  global gRoundOver, gsoundFrame, gEgger  
  if gRoundOver = false then  
    putUpEgg gEgger  
    go to frame gsoundFrame  
  else  
    putUpEgg gEgger  
    go to frame "QuitOrGoOn"  
    set gRoundOver = false  
  end if  
end
```

Score Script7

```
on exitFrame  
  checkuserTimeOut  
  go to the frame  
end
```

Score Script8

```
on exitFrame
  -- use gLevel to figure out which game we're playing
  global glevel
  if gLevel <= 30 then
    go to frame "VowelSound"
  else
    go to frame "CVsound"
  end if
end
```

Score Script9

```
on exitFrame
  go to frame "CVtest"
end
```

Score Script10

```
on exitFrame
  global gVowelTest, gSoundFrame, gHandCursor
  put Label(0) into gSoundFrame
  doVowelTest gVowelTest
  repeat while soundbusy(1)
    nothing
  end repeat
  SetUserTimeout (600, "vowelTestTimeout")
  cursor gHandcursor
  go to frame "VowelTest"
end
```

Score Script18

```
on mouseUp
  global gVowelTest, gPauseStatus
  if gPauseStatus = "resume" then exit
  CheckForTimeOut gVowelTest
  checkanswer (gVowelTest, #same)
end
```

Score Script19

```
on mouseUp
  global gVowelTest, gPauseStatus
  if gPauseStatus = "resume" then exit
  checkforTimeOut gVowelTest
  checkanswer (gVowelTest, #diff)
end
```

Score Script20

```
on exitFrame
  global gCVtest, gSoundFrame, gHandcursor
  put Label(0) into gSoundFrame
  doCVTest gCVtest
  repeat while soundbusy(1)
    nothing
  end repeat
  SetUserTimeout (600, "CVTestTimeOut")
  cursor gHandcursor
  go to frame "CVtest"
end
```


Movie Script21

```
on startMovie
  global gVowelSounds, gCVTest, gVowelTest, gRoundOver, gspritesOnList,
  gEgger, gpausestatus
  Global gEggBasketSprites, gOldColorLevel, goldscoringlevel, gthename, gRecordKeeper
  global gHandCursor, goldscoringlist

  cursor 4
  -- if the machineType <> 256 then
  --   set gOldColorLevel = the colorDepth
  --   set the colorDepth = 8
  -- end if
  set gCVTest = new(script "CVTestProctor")
  set x = the number of lines of field "vowels"
  set gVowelSounds = []
  repeat with z = 1 to x
    set temp = []
    setat(temp,1,item 1 of line z of field "vowels")
    setat(temp,2,item 2 of line z of field "vowels")
    append gVowelSounds, temp
  end repeat
  set gVowelTest = new(script "VowelTest")
  -- set gEgger = new(script "egger")
  set gRoundOver = false
  set gspritesOnList = []
  set gEggBasketSprites = {11,12,13,14,15,16,17,18,19,20}
  repeat with thisSprite in gEggBasketSprites
    set the visible of sprite thisSprite = false
  end repeat
  updatestage

  -- go to frame "preStart"

  set the visible of sprite 45 to false
  set the visible of sprite 46 to false
  set the visible of sprite 47 to false
  set gpausestatus = "pause"
  -- if objectP(gRecordKeeper) then
  --   put setUpRound (gRecordKeeper, gTheName, 2) into glistscoringlevel
  -- end if
  --
  -- put getat(glistscoringlevel,1) into goldscoringlevel

  if objectP(gRecordKeeper) then
    put setUpRound (gRecordKeeper, gTheName, 2) into goldscoringlevel
    put getGameLevelLists(gRecordKeeper,gthename) into templevellist
    put getat(templevellist,2) into goldscoringlist
    put "goldscoringlist =" & goldscoringlist
    if ListP(goldscoringlevel) then -- we are playing the second game

      set goldscoringlevel = getat(goldscoringlevel,1)
      if goldscoringlevel = 115 then set goldscoringlevel = 114
    end if
  end if
```

```

set gHandCursor = []
append gHandCursor, member "HandCur"
append gHandCursor, member "HandCurMask"
set the purgePriority of member "HandCur" = 0
set the purgePriority of member "HandCurMask" = 0
end

on StopMovie
    global gVowelSounds, gCVTest, gVowelTest, gVoid, gLevel, gspritesOnList, gWhichEgg,
    gEgger
    global gBounceNum, gEggBasketSprites, gOldColorLevel, gHandCursor
    -- if the machineType <> 256 then
    --     set the colorDepth = gOldColorLevel
    -- end if
    if the runMode <> "author" then
        set gCVTest = gVoid
        set gVowelSounds = gVoid
        set gVowelTest = gVoid
        set gEgger = gVoid
    end if
    set gRoundOver = gVoid
    set gLevel = gVoid
    set gspritesOnList = gVoid
    set gWhichEgg = gVoid
    set gBounceNum = gVoid
    set gEggBasketSprites = gVoid
    set gHandCursor = gVoid
    set the purgePriority of member "HandCur" = 3
    set the purgePriority of member "HandCurMask" = 3
end

```

Parent Script22:CVTestProctor

--4/8/97

--EggBasket Replacement

Property TestPairList, ancestor, myHandlers, currentLevel, DiffList, SameList
property roundScoreList, SoundGroup, Answer, NumRightInARow, NumWrongInARow
Property LevelToSave, PairToPlay, timeoutNum, TimeOutAnswer
Property disableReplayButton, skipYesOrNo

on x-----Public Handlers -----
-- I'm a separator
end

on new me
global gRecordKeeper
if ObjectP(gRecordKeeper) then
set the ancestor of me to gRecordKeeper
set myHandlers = 0
set myHandlers = GetMyHandlers(me)
end if
return Me
end

on startNewRound me, whichLevel, whichsoundGroup, PrefList
-- Pub.
-- sets up lists of soundpairs for current group of levels
-- "WhichSoundGroup" is a string which all the sounds for this
-- group share with only numerical suffixes changing
-- the soundGroup property never changes during a round
global gRoundOver
setUpTestPairList me
set RoundScorelist = [:]
set soundgroup = whichsoundGroup
initDiffSameLists me
set numRightInARow = 0
set NumWrongInARow = 0
set currentlevel = whichLevel
set LevelToSave = currentLevel
set timeoutNum = 0
set TimeOutanswer = empty
set gRoundOver = false
set disableReplayButton = getAT(PrefList,1)
set skipYesOrNo = getAT(PrefList,2)
put soundGroup
end

on doSampleSounds me, whichSounds
-- pub.
-- called from the score (ultimately), plays a pair of
-- same sounds if "whichSounds" = #same or different sounds
-- if "whichSounds" = #diff. Sounds are pulled from the
-- current sound group
if whichsounds = #same then
set sound1 = soundGroup&1
set samplePair = {sound1,sound1}
else
set sound1 = soundGroup&1
set sound2 = soundgroup&9
end if

```

    set samplePair = [sound1,sound2]
end if
set firstSound = getat(samplePair,1)
set secondSound = getat(samplePair,2)
puppetsound firstSound
updatetage
repeat while soundBusy(1)
    nothing
end repeat
puppetsound 0
wait 30
puppetsound secondSound
updatetage
repeat while soundBusy(1)
    nothing
end repeat
puppetsound 0
end

on DoCVTest me
-- pub...
-- this is handler to play test
-- it plays the two sounds and sets the property "answer" according to
-- whether the answer is same or different. These options are passed as
-- symbols, ie. #same or #diff
set level = (currentlevel-30)mod(7)
if level = 0 then set Level = 7
set PlaysLeft = count(testPairList)
if PlaysLeft < 1 then exit
set NextPair = getat(testPairList, playsLeft)
deleteAt (testPairList, playsLeft)
if nextPair = #Diff then
    set pairList = getprop(difflist,Level)
    set pair = random(count(PairList))
    set PairToPlay = getAt(PairList,Pair)
    set Answer = #Diff
else
    set pairList = getprop(samelist,Level)
    set pair = random(count(PairList))
    set PairToPlay = getAt(PairList,Pair)
    set answer = #same
end if
put answer
set sound1 = getat(PairToPlay,1)
puppetsound member sound1
updatetage
repeat while soundbusy(1)
    nothing
end repeat
puppetsound 0
wait 30
set sound2 = getat(PairToPlay,2)
puppetsound member sound2
updatetage
repeat while soundbusy(1)
    nothing
end repeat
puppetsound 0

put the name of member sound1
put the name of member sound2

```

```

put "current level = " & currentLevel
put "levelToSave = " & levelToSave
put roundScoreList
end

```

```

on RepeatStimuli me

```

```

-- pub.
-- replays two current sounds if
-- user needs to
set sound1 = getat(PairToPlay,1)
set sound2 = getat(PairToPlay,2)
puppetsound member sound1
updatestage
repeat while soundbusy(1)
  nothing
end repeat
puppetsound 0
wait 30
puppetsound member sound2
updatestage
repeat while soundbusy(1)
  nothing
end repeat
puppetsound 0

```

```

end

```

```

on CheckAnswer me, whichAnswer

```

```

--this handler should be called from the
-- choice buttons, ie the hen pairs on screen.
-- it evaluates the answer given by the user and keeps a property list
-- accordingly to report to the RecordKeeper. The answers must be passed
-- as symbols #Same or #diff. The list being kept is a property list
-- with the current level as the property, the value is itself a list
-- with 4 items, the first 2 are for scores on different sounds, the
-- last 2 are for scores on same sounds.
-- This handler also keeps track of how many right or wrong in a row
-- the user answers during a round. If 4 right answers in a row then
-- the level is raised by 1, if 2 wrong in a row it drops by 1.

```

```

global gRoundOver, gWhichEgg

```

```

if whichanswer = answer then --Right!!

```

```

  set gWhichEgg = "Correct Egg"
  set NumRightInARow = NumRightInARow + 1
  set NumWrongInARow = 0
  if answer = #diff then

```

```

    hideReplayButton me

```

```

    go to frame "Diff correct"

```

```

    set AnsList = getaProp(roundscoreList,currentlevel)

```

```

    if voidP(AnsList) then -- no score at this level yet

```

```

      set AnsList = [1,1,0,0]

```

```

      addProp roundscorelist, currentLevel, ansList

```

```

    else

```

```

      set NumPlays = getat (ansList, 1)

```

```

      set NumPlays = numplays + 1

```

```

      set NumRight = getAt(ansList,2)

```

```

      set Numright = numright + 1

```

```

      setAt AnsList, 1, numPlays

```

```

      setAt AnsList, 2, numRight

```

```

      setaProp RoundscoreList,CurrentLevel, anslist

```

```

    end if

```

```

else -- answer was #same
  hideReplayButton me
  go to frame "Same Correct"
  set AnsList = getaProp(roundscoreList,currentlevel)
  if voidP(AnsList) then -- no score at this level yet
    set AnsList = [0,0,1,1]
    addProp roundscorelist, currentLevel, ansList
  else
    set NumPlays = getat (ansList, 3)
    set NumPlays = numplays + 1
    set NumRight = getAt(ansList,4)
    set Numright = numright + 1
    setAt AnsList, 3, numPlays
    setAt AnsList, 4, numRight
    setaProp RoundscoreList,CurrentLevel, anslist
  end if
end if

else -- wrong!!
  set gWhichEgg = "wrong Egg"
  set NumRightinaRow = 0
  set NumWronginaRow = NumwronginaRow + 1

  if answer = #diff then
    hideReplayButton me
    go to frame "same Wrong"
    set AnsList = getaProp(roundscoreList,currentlevel)
    if voidP(AnsList) then -- no score at this level yet
      set AnsList = [1,0,0,0]
      addProp roundscorelist, currentLevel, ansList
    else
      set NumPlays = getat (ansList, 1)
      set NumPlays = numplays + 1
      setAt AnsList, 1, numPlays
      setaProp RoundscoreList,CurrentLevel, anslist
    end if
  else
    hideReplayButton me
    go to frame "diff Wrong"
    set AnsList = getaProp(roundscoreList,currentlevel)
    if voidP(AnsList) then -- no score at this level yet
      set AnsList = [0,0,1,0]
      addProp roundscorelist, currentLevel, ansList
    else
      set NumPlays = getat (ansList, 3)
      set NumPlays = numplays + 1
      setAt AnsList, 3, numPlays
      setaProp RoundscoreList,CurrentLevel, anslist
    end if
  end if

end if
if numrightinarow > 3 then
  if (currentlevel-30)mod(7) = 0 then -- we are at level 7 within a group
    set currentlevel = currentlevel
    set LeveltoSave = CurrentLevel + 1
  else
    set currentLevel = currentlevel + 1
    set LeveltoSave = currentLevel
  end if
end if

```



```

    end if
    set NumRightInARow = 0
end if
if numWrongInARow > 1 then
    if (currentLevel-30)mod(7) = 1 then
        set currentLevel = currentlevel
        set levelToSave = currentlevel
    else
        set currentlevel = currentlevel - 1
        set levelToSave = currentlevel
    end if
    set numWrongInARow = 0
end if
if count(testPairList) < 1 then
    set gRoundOver = true
    exit
end if
end

```

```

on doTimeout me
    global gWhichEgg, gRoundOver
    if count(testPairList) < 1 then -- end of round so set flag
        set gRoundOver = true
    end if

    if TimeoutNum = 1 then -- second timeout in a row so bailout
        hideReplayButton me
        go to Label("playAgain?") + 1 -- skip prefs test and go straight to prompt
        if count(roundscoreList) > 0 then
            -- user played some in this round so report scores
            doEndOfRound me
        end if
        set timeoutNum = 0
        set TimeoutAnswer = empty
    else
        -- first adjust current level for wrong answer
        set NumWrongInARow = NumWrongInARow + 1
        set NumRightInARow = 0
        if numWrongInARow > 1 then
            if (currentLevel-30)mod(7) = 1 then
                set currentLevel = currentlevel
            else
                set currentlevel = currentlevel - 1
            end if
            set numWrongInARow = 0
        end if
        if gRoundOver = false then -- not last egg
            -- do egg drop and store scores to report
            -- if user plays again. else bailout above happens
            set gWhichEgg = "Wrong egg"
            set timeoutNum = TimeoutNum + 1
            Set TimeoutAnswer = answer
            if answer = #Diff then
                hideReplayButton me
                go to frame "Diff Drop"
            else
                hideReplayButton me
            end if
        end if
    end if
end

```



```

    go to frame "same Drop"
end if
else -- timeout on last egg so do scoring here
    -- add scores to LevelToSave since currentLevel has changed to
    -- reflect the 2 wrong scores in a row
    set gWhichEgg = "Wrong egg"
    set AnsList = getaProp(roundscoreList, levelToSave)
    if voidP(ansList) then set AnsList = [0,0,0,0] -- no scores yet at this level
    if answer = #Diff then
        hideReplayButton me
        go to frame "Diff Drop"
        set NumPlays = getat (ansList, 1)
        set NumPlays = numplays + 1
        setAt AnsList, 1, numPlays
    else
        hideReplayButton me
        go to frame "same Drop"
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        setAt AnsList, 3, numPlays
    end if
    setaProp roundscoreList, LevelToSave, anslist
    set leveltosave = currentLevel

end if
end if
end

on CheckForTimeout me
    -- call from "answer chickens" on stage
    -- resets timeout and reports timeout score
    if TimeoutNum > 0 then -- there is one Timeout already so adjust score for resumed play
        set timeoutNum = 0 -- reset counter
        -- add saved scores to leveltoSave (level played at the time Out) not
        -- to current level which may have changed
        -- set AnsList = getaProp(roundscoreList, currentlevel)
        set AnsList = getaProp(roundscoreList, levelToSave)
        -- retrieve list to report score from timeout egg
        if voidP(ansList) then set AnsList = [0,0,0,0] -- no scores yet at this level
        if Timeoutanswer = #diff then
            set NumPlays = getat (ansList, 1)
            set NumPlays = numplays + 1
            setAt AnsList, 1, numPlays
        else
            set NumPlays = getat (ansList, 3)
            set NumPlays = numplays + 1
            setAt AnsList, 3, numPlays
        end if
        -- setaProp roundscoreList, currentLevel, anslist
        setaProp roundscoreList, levelToSave, anslist
        set leveltosave = currentLevel -- readjust levelToSave here, not on timeout.
    end if
end

on doEndOfRound me
    if not objectP(gRecordKeeper) then exit
    if voidP(leveltosave) then set leveltosave = currentLevel
    if leveltosave > 115 then set leveltosave = 115

```

```

put roundscoreList
put "LevelToSave" =&& levelToSave
saveroundscores(me,roundscorelist,leveltosave)
end

```

```

on xx-----Private Handlers-----
-- i'm a separator
end

```

```

on InitDiffSameLists me
set s1 = soundgroup&1
set s1 = the number of member s1
set s2 = soundgroup&2
set s2 = the Number of member s2
set s3 = soundgroup&3
set s3 = the Number of member s3
set s4 = soundgroup&4
set s4 = the Number of member s4
set s5 = soundgroup&5
set s5 = the Number of member s5
set s6 = soundgroup&6
set s6 = the Number of member s6
set s7 = soundgroup&7
set s7 = the Number of member s7
set s8 = soundgroup&8
set s8 = the Number of member s8
set s9 = soundgroup&9
set s9 = the Number of member s9

set DiffList = []
set SameList = []
repeat with x = 1 to 7
case (x) of
1: addProp DiffList ,1, [[s1,s9],[s9,s1]]
addProp samelist ,1, [[s1,s1],[s9,s9]]
2: addProp DiffList ,2, [[s1,s8],[s8,s1],[s2,s9],[s9,s2]]
addProp samelist ,2, [[s1,s1],[s2,s2],[s8,s8],[s9,s9]]
3: addProp DiffList ,3, [[s1,s7],[s7,s1],[s2,s8],[s8,s2],[s3,s9],[s9,s3]]
addProp samelist ,3, [[s1,s1],[s2,s2],[s3,s3],[s7,s7],[s8,s8],[s9,s9]]
4: addProp DiffList ,4,
[[s1,s6],[s6,s1],[s2,s7],[s7,s2],[s3,s8],[s8,s3],[s4,s9],[s9,s4]]
addProp samelist ,4,
[[s1,s1],[s2,s2],[s3,s3],[s4,s4],[s6,s6],[s7,s7],[s8,s8],[s9,s9]]
5: addProp DiffList ,5, [[s2,s6],[s6,s2],[s3,s7],[s7,s3],[s4,s8],[s8,s4]]
addProp samelist ,5, [[s2,s2],[s3,s3],[s4,s4],[s6,s6],[s7,s7],[s8,s8]]
6: addProp DiffList ,6, [[s3,s6],[s6,s3],[s4,s7],[s7,s4]]
addProp samelist ,6, [[s3,s3],[s4,s4],[s6,s6],[s7,s7]]
7: addProp DiffList ,7, [[s4,s6],[s6,s4]]
addProp samelist ,7, [[s4,s4],[s6,s6]]
end case
end repeat
end

```

```

on setUpTestPairList me
set testpairlist to []
repeat with x = 1 to 10
append testpairlist , #Diff
end repeat
set TempList = [1,2,3,4,5,6,7,8,9,10]
repeat with x = 1 to 6

```

```

    set y = count(templist)
    deleteat templist, random(y)
end repeat
repeat with pos in templist
    setAt(testpairlist, pos, #same)
end repeat
put testpairlist
end

on hideReplayButton me
    -- hides the replay button if it is indeed showing
    if disableReplayButton = true then exit -- no need to hide, since it's not on stage
    set the loc of sprite 25 = point(-1000, -1000)
    set the loc of sprite 26 = point(-1000, -1000)
    updatestage
    puppetSprite 25, false
    puppetSprite 26, false
end

on xxx-----Testing Handlers-----
    -- i'm a separator
    nothing
end

on showhandlers me
    put myhandlers
end

on showProps me
    -- testing
    -- puts list of properties and their current values in message window
    set PropNum = count(me)
    repeat with x = 1 to PropNum
        set prop = 0
        set thisProp = getpropat(me, x)
        if thisProp = #myHandlers then next repeat
        put (string (getpropat(me, x))) &&"&& getaProp(me, thisProp) into prop
        put prop
    end repeat
end

```

Score Script23

```

on mouseUp
    global gCVtest, gPauseStatus
    if gPauseStatus = "resume" then exit
    CheckForTimeout gCVtest
    checkanswer (gCVtest, #same)
end

```

Score Script24

```
on mouseUp  
  global gCVtest.gPauseStatus  
  if gPauseStatus = "resume" then exit  
  checkForTimeout gCVtest  
  checkanswer (gCVtest, #Diff)  
end
```

Score Script25

```
on exitFrame  
  cursor 200  
  spritesOn (30,39)  
end
```

Script of Cast Member26

```
on mouseUp  
  global gCVTest  
  checkAnswer (gCVTest, #Diff)  
end
```

Script of Cast Member27

```
on mouseUp  
  global gCVtest  
  doCVtest (gCVtest, )  
end
```

Score Script28

```
on exitFrame
-- global gRecordKeeper, gTheName, gLevel, gBounceNum
-- if objectP(gRecordKeeper) then
--   put setUpRound (gRecordKeeper, gTheName, 2) into glevel
--
--   if ListP(gLevel) then -- we are playing the second game
--     set gBounceNum = getat(gLevel,2)
--     set gLevel = getat(gLevel,1)
--   end if
-- else
--   set gLevel = 1 -- change here to integer over 30 to play game 2
--   set gBounceNum = 0
-- end if
spriteListOff
```

end

```

on ConvertScoringLevel scoringLevel, prefList
  -- take stored level and sees first if the level
  -- plays at "vowel" or "CV" game.
  -- If "CV" it pulls out correct SoundFamily prefix.
  global gCVtest, gVowelTest, gBounceNum
  if scoringLevel <= 30 then -- we are playing vowel game
    startnewround (gVowelTest, scoringlevel, prefList)
    -- go to frame "Vowelsound"
  else
    set scoringlevel = value(scoringLevel)
    if gBounceNum = 5 then -- change here to set different bounce number
      -- we need to jump to next soundFamily
      -- take scoringLevel and subtract out 30 vowel levels
      -- divide by 7 (make sure it's an integer!!)
      -- then add one and times by 7 to jump a level (add 31 too!)
      set scoringLevel = (scoringLevel-30)/7
      set scoringLevel = ((scoringLevel+1)*7)+31
    end if
    case (true) of
      ((31<=(scoringLevel))and((scoringLevel)<=37)):set soundFamily = "RL165df"
      ((38<=(scoringLevel))and((scoringLevel)<=44)):set soundFamily = "RL165nr"
      ((45<=(scoringLevel))and((scoringLevel)<=51)):set soundFamily = "RL110df"
      ((52<=(scoringLevel))and((scoringLevel)<=58)):set soundFamily = "RL110nr"
      ((59<=(scoringLevel))and((scoringLevel)<=65)):set soundFamily = "DG80DF"
      ((66<=(scoringLevel))and((scoringLevel)<=72)):set soundFamily = "DG80nr"
      ((73<=(scoringLevel))and((scoringLevel)<=79)):set soundFamily = "DG40df"
      ((80<=(scoringLevel))and((scoringLevel)<=86)):set soundFamily = "DG40nr"
      ((87<=(scoringLevel))and((scoringLevel)<=93)):set soundFamily = "mn80df"
      ((94<=(scoringLevel))and((scoringLevel)<=100)):set soundFamily = "mn80nr"
      ((101<=(scoringLevel))and((scoringLevel)<=107)):set soundFamily = "mn40df"
      ((108<=(scoringLevel))and((scoringLevel)<=114)):set soundFamily = "mn40nr"
    end case
    startnewround (gCVtest, ScoringLevel, soundFamily, PrefList)
    -- set gStartFrame = "CVsound" -- store which game to go to
  end if
end

on ConvertJimScoringLevel scoringLevel
  -- take stored level and sees first if the level
  -- plays at "vowel" or "CV" game.
  -- If "CV" it pulls out correct SoundFamily prefix.
  global gCVtest, gVowelTest, gBounceNum
  if scoringLevel <= 30 then
    startnewround (gVowelTest, scoringlevel)
    -- go to frame "Vowelsound"
  else
    set scoringlevel = value(scoringLevel)
    if gBounceNum = 5 then
      -- we need to jump to next soundFamily
      -- take scoringLevel and subtract out 30 vowel levels
      -- divide by 7 (make sure it's an integer!!)
      -- then add one and times by 7 to jump a level (add 31 too!)
      set scoringLevel = (scoringLevel-30)/7
      set scoringLevel = ((scoringLevel+1)*7)+31
    end if
    case (true) of
      ((31<=(scoringLevel))and((scoringLevel)<=37)):set soundFamily = "DG80DF"
      ((38<=(scoringLevel))and((scoringLevel)<=44)):set soundFamily = "DG80nr"

```



```

    ((45<=(scoringLevel))and((scoringLevel)<=51)):set soundFamily = "DG40df"
    ((52<=(scoringLevel))and((scoringLevel)<=58)):set soundFamily = "DG40nr"
    ((59<=(scoringLevel))and((scoringLevel)<=65)):set soundFamily = "RL165df"
    ((66<=(scoringLevel))and((scoringLevel)<=72)):set soundFamily = "RL165nr"
    ((73<=(scoringLevel))and((scoringLevel)<=79)):set soundFamily = "RL110df"
    ((80<=(scoringLevel))and((scoringLevel)<=86)):set soundFamily = "RL110nr"
    ((87<=(scoringLevel))and((scoringLevel)<=93)):set soundFamily = "mn80df"
    ((94<=(scoringLevel))and((scoringLevel)<=100)):set soundFamily = "mn80nr"
    ((101<=(scoringLevel))and((scoringLevel)<=107)):set soundFamily = "mn40df"
    ((108<=(scoringLevel))and((scoringLevel)<=114)):set soundFamily = "mn40nr"
end case
startnewround (gCVtest, ScoringLevel, soundFamily)
-- set gStartFrame = "CVsound"
end if
end

```

Movie Script30

```

on stripDots whichCastLib
    repeat with x = 11 to 12
        -- repeat with x = 1 to whichCast of castlib whichcastLib
        put the name of member x of castlib whichcastLib into temp
        set Y = offset (".",temp)
        set temp = char 1 to (y-1) & temp
        set the name of member x of castlib whichcastLib = temp
    end repeat
end

```

Parent Script31:vowelTest

--4/9/97

--EggBasket Replacement

property TestPairList, currentLevel, RoundList, ancestor, myHandlers, answer
property pairToPlay, TimeOutNum, TimeOutanswer, sampleSoundList
Property disableReplayButton, skipYesOrNo

on x-----Public Handlers -----
-- I'm a separator
end

on new me
global gRecordKeeper
if objectP(gRecordKeeper) then
set the ancestor of me to gRecordKeeper
set myHandlers = 0
set myHandlers = GetMyHandlers(me)
end if
return me
end

on startNewRound me, level, PrefList
global gRoundOver
set CurrentLevel = level
setUpVowelList me, currentLevel
set RoundList = [:]
addProp (roundList, currentLevel, [0,0,0,0])
set TimeOutNum = 0
set gRoundOver = false
set disableReplayButton = getAT(prefList,1)
set skipYesOrNo = getAT(prefList,2)
end

on doSampleSounds me, whichSounds
-- pub.
-- called from the score (ultimately), plays two pairs of
-- same sounds if "whichSounds" = #same or different sounds
-- if "whichSounds" = #diff. Sounds are stored in two lists in property
-- SampleSoundList, first sublist are same, second are different
if whichsounds = #same then
set samplePairs = [getat(sampleSoundList,1),getat(sampleSoundList,2)]
else
set samplePairs = [getat(sampleSoundList,3),getat(sampleSoundList,4)]
end if
repeat with samplePair in samplePairs
set firstSound = getat(samplePair,1)
set secondSound = getat(samplePair,2)
puppetsound member firstSound
updatestage
repeat while soundBusy(1)
nothing
end repeat
puppetsound 0
wait 30
puppetsound member secondSound
updatestage
repeat while soundBusy(1)

```

    nothing
  end repeat
  puppetsound 0
  wait 90
end repeat
end

on doVowelTest me
  set PlaysLeft = count(testPairList)
  if PlaysLeft < 1 then exit
  set pairToPlay = getat(testPairList, playsLeft)
  deleteAt (testPairList, playsLeft)
  set sound1 = getat(pairToPlay,1)
  set sound2 = getat(pairToPlay,2)
  if sound1 = sound2 then
    set answer = #same
  else
    set answer = #diff
  end if
  puppetsound sound1
  updatetage
  repeat while soundbusy(1)
    nothing
  end repeat
  puppetsound 0
  wait 30
  puppetsound sound2
  updatetage
  repeat while soundbusy(1)
    nothing
  end repeat
  puppetsound 0

  put answer
  put the name of member sound1
  put the name of member sound2
end

```

```

on RepeatStimuli me
  set sound1 = getat(pairToPlay,1)
  set sound2 = getat(pairToPlay,2)
  puppetsound sound1
  updatetage
  repeat while soundbusy(1)
    nothing
  end repeat
  puppetsound 0

  wait 30
  puppetsound sound2
  updatetage
  repeat while soundbusy(1)
    nothing
  end repeat
  puppetsound 0

```

end

```

on checkAnswer me, whichAnswer
  global gSoundOver, gWhichEgg

```

```

set AnsList = getaProp(roundList,currentlevel)
if whichAnswer = answer then -- right
    set gWhichEgg = "Correct Egg"
    if answer = #diff then
        set NumPlays = getat (ansList, 1)
        set NumPlays = numplays + 1
        set NumRight = getat(ansList,2)
        set Numright = numright + 1
        setAt AnsList, 1, numPlays
        setAt AnsList, 2, numRight
        hideReplayButton me
        go to frame "diff Correct"
    else -- answer was #same
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        set NumRight = getat(ansList,4)
        set Numright = numright + 1
        setAt AnsList, 3, numPlays
        setAt AnsList, 4, numRight
        hideReplayButton me
        go to frame "same Correct"
    end if
else -- wrong!!
    set gWhichEgg = "wrong Egg"
    if answer = #diff then
        set NumPlays = getat (ansList, 1)
        set NumPlays = numplays + 1
        setAt AnsList, 1, numPlays
        hideReplayButton me
        go to frame "same wrong"
    else
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        setAt AnsList, 3, numPlays
        hideReplayButton me
        go to frame "diff wrong"
    end if
end if
setaProp RoundList,CurrentLevel, anslist
if count(testPairList) < 1 then

    set gRoundOver = true
    exit
end if
end

on doEndOfRound me
    if not objectP(gRecordKeeper) then exit
    set scores = getProp(roundList, currentLevel)
    if (getat(scores,2) >= 5) and (getat(scores,4) = 4) then
        set levelToSave = currentLevel + 1
    else
        set levelToSave = currentLevel
    end if
    -- put "roundList = "&roundList
    -- put "levelToSave = "& levelToSave
    SaveRoundScores (me, roundlist, levelToSave)
end

```

```

on doTimeout me
  global gWhichEgg, gRoundOver
  if count(testPairList) < 1 then
    set gRoundOver = true
  end if
  if TimeoutNum = 1 then
    hideReplayButton me
    go to Label( "playAgain?" ) + 1 -- skip prefs test and go straight to prompt
    -- go to frame "PlayAgain?"
    set AnsList = getaProp(roundList,currentlevel)
    if getat(anslist,1) > 0 or getat(anslist,3) > 0 then
      -- user played some in this round
      doEndOfRound me
    end if
    set the timeoutscript = empty
    set TimeoutNum = 0
    set TimeoutScore = empty
  else
    if gRoundOver = false then -- not last egg
      set gWhichEgg = "Wrong egg"
      set TimeoutNum = TimeoutNum + 1
      set Timeoutanswer = answer
      if answer = #Diff then
        hideReplayButton me
        go to frame "Diff Drop"
      else
        hideReplayButton me
        go to frame "same Drop"
      end if
    else -- timeout on last egg!!
      set gWhichEgg = "Wrong egg"
      set AnsList = getaProp(roundList,currentlevel)
      if answer = #diff then
        hideReplayButton me
        go to frame "Diff Drop"
        set NumPlays = getat (ansList, 1)
        set NumPlays = numplays + 1
        setAt AnsList, 1, numPlays
      else
        hideReplayButton me
        go to frame "same Drop"
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        setAt AnsList, 3, numPlays
      end if
    end if
  end if
end if
end

```

```

on CheckForTimeout me
  -- call from "answer chickens" on stage
  -- resets timeout and reports timeoutscore
  if TimeoutNum > 0 then
    set TimeoutNum = 0
    set AnsList = getaProp(roundList,currentlevel)
    if Timeoutanswer = #diff then
      set NumPlays = getat (ansList, 1)
      set NumPlays = numplays + 1
    end if
  end if
end

```

```

    setAt AnsList, 1, numPlays
else
    set NumPlays = getAt (ansList, 3)
    set NumPlays = numPlays + 1
    setAt AnsList, 3, numPlays
end if
end if
end

on xx-----Private Handlers-----
-- i'm a separator
end

on hideReplayButton me
-- hides the replay button if it is indeed showing
if disableReplayButton = true then exit -- no need to hide, since it's not on stage
set the loc of sprite 25 = point(-1000,-1000)
set the loc of sprite 26 = point(-1000,-1000)
updateStage
puppetSprite 25, false
puppetSprite 26, false
end

on setUpVowelList me, whichLevel
-- sets up list of 10 lists, each sublist contains two items,
-- the names of sounds from a global list "gVowelSounds"
-- these sounds are either paired with themselves or are
-- paired with the other sound. As per instruction, the handler sets up
-- the ten list randomly with 6 lists of "different" pairs and
-- 4 lists of "same" pairs
global gVowelSounds
if whichLevel > count(gVowelSounds) or whichLevel < 1 then
    alert "There are no vowel sounds at that level. Check your lingo."
    abort
end if
set soundlist = getAt(gVowelSounds, whichLevel)
repeat with x = 1 to 2 -- change names to cast numbers
    set Sound = getAt(soundList, x)
    set sound = the member of member sound
    setAt soundlist, x, sound
end repeat
set xx = [getAt(soundlist,1),getAt(soundlist,1)]
set xy = [getAt(soundlist,1),getAt(soundlist,2)]
set yy = [getAt(soundlist,2),getAt(soundlist,2)]
set yx = [getAt(soundlist,2),getAt(soundlist,1)]
set VowelPairList = []
set sameList = [xx,yy]
set diffList = [xy,yx]

set sampleSoundList = [xx,yy,xy,yx]
-- for use during "doSampleSounds" handler
-- first two items are lists of two same sounds
-- second two are lists of two different sounds

repeat with x = 1 to 10
    append VowelPairList, getAt(diffList, random(2))
end repeat

```

```

set TempList = [1,2,3,4,5,6,7,8,9,10]
repeat with x = 1 to 6
    set y = count(tempList)
    deleteat tempList, random(y)
end repeat
repeat with pos in tempList
    setAt(VowelPairList,pos, getAt(samelist,random(2)))
end repeat
set testPairList = vowelPairList
end

on xxx-----Testing Handlers-----
    -- i'm a separator
    nothing
end

on showhandlers me
    put myhandlers
end

on showProps me
    -- testing
    -- puts list of properties and their current values in message window
    set PropNum = count(me)
    repeat with x = 1 to PropNum
        set prop = 0
        set thisProp = getpropat(me, x)
        if thisProp = #myHandlers then next repeat
        put (string (getpropat(me, x))) &&"=" && getaProp(me,thisProp) into prop
        put prop
    end repeat
end
end

```



```

on setUpVowelList whichLevel
  -- sets up list of 10 lists. each sublist contains two items,
  -- the names of sounds from a global list "gVowelSounds"
  -- these sounds are either paired with themselves or are
  -- paired with the other sound. As per instruction, the handler sets up
  -- the ten list randomly with 6 lists of "different" pairs and
  -- 4 lists of "same" pairs
  global gVowelSounds
  if whichLevel > count(gVowelSounds) or whichLevel < 1 then
    alert "That level is out of range. Check your lingo"
    abort
  end if
  set soundlist = getat(gvowelsounds, whichLevel)
  repeat with x = 1 to 2 -- change names to cast numbers
    set Sound = getat(soundList, x)
    set sound = the member of member sound
    setat soundlist, x, sound
  end repeat
  set xx = [getat(soundlist,1),getat(soundlist,1)]
  set xy = [getat(soundlist,1),getat(soundlist,2)]
  set yy = [getat(soundlist,2),getat(soundlist,2)]
  set yx = [getat(soundlist,2),getat(soundlist,1)]
  set VowelPairList = []
  set sameList = [xx,yy]
  set diffList = [xy,yx]
  repeat with x = 1 to 10
    append VowelPairList, getat(difflist, random(2))
  end repeat
  set TempList = {1,2,3,4,5,6,7,8,9,10}
  repeat with x = 1 to 6
    set y = count(templist)
    deleteat templist, random(y)
  end repeat
  repeat with pos in tempList
    setAt(VowelPairList,pos, getat(samelist,random(2)))
  end repeat
  put VowelPairList
  return VowelPairList
end

```

Movie Script33

```
on putUpEgg whichEgg
  put member "no egg" into NoEgg
  put member whichegg into thisEgg
  repeat with x = 30 to 39
    if the member of sprite x <> NoEgg then
      next repeat
    else
      set the member of sprite x = thisEgg
      updatestage
      exit
    end if
  end repeat
end
```

Score Script34

```
on mouseUp
  global gVowelTest
  CheckAnswer (gVowelTest, #same)
end
```

Parent Script35:Egger

```
property EggSprites, sameWrongsprite, DiffWrongsprite, sameWrongNum, DiffWrongNum, RightNum
```

```
on new me
```

```
    set sameWrongsprite = 4
    set DiffWrongsprite = 5
    puppetsprite sameWrongsprite, true
    puppetsprite DiffWrongsprite, true
    init me
    return me
```

```
end
```

```
on PutUpEgg me
```

```
    global gWhichEgg
    if count(eggSprites) < 1 then exit
    set thisEgg = getat(eggSprites, 1)
    deleteat(eggSprites, 1)
    set the member of sprite thisEgg = member gWhichEgg
    updatestage
```

```
end
```

```
on Init me
```

```
    global gEggBasketSprites
    set eggSprites = [30,31,32,33,34,35,36,37,38,39]
    repeat with x = 10 down to 1
        set the member of sprite getat(eggSprites,x) = member "no Egg"
        updatestage
        wait 4
    end repeat
    set sameWrongNum = 0
    set DiffWrongNum = 0
    set RightNum = 0
    set the member of sprite sameWrongsprite = member "SameBroke 0"
    set the member of sprite DiffWrongsprite = member "DiffBroke 0"
    repeat with thisSprite in gEggBasketSprites
        set the visible of sprite thisSprite = false
    end repeat
    updatestage
```

```
end
```

```
on putUpBrokenEgg me, where
```

```
    unloadmember
    if where = #diff then
        set DiffWrongNum = DiffWrongNum + 1
        set nextEgg = "DiffBroke"&& DiffWrongNum
        set the member of sprite DiffWrongsprite = member nextEgg
    else
        set sameWrongNum = sameWrongNum + 1
        set nextEgg = "sameBroke"&& sameWrongNum
        set the member of sprite sameWrongsprite = member nextEgg
    end if
    preloadmember "crack"
    puppetsound 2, "crack"
    updatestage
```

```
end
```

```
on doEggBasket me
  global gEggBasketSprites
  set RightNum = RightNum + 1
  set the visible of sprite getat(gEggbasketSprites, rightNum) = true
  updatestage
end
```

Score Script36

```
on mouseUp
  global gVowelTest
  CheckAnswer (gVowelTest, #Diff)
end
```

Score Script37

```
on mouseUp
  global gVowelTest
  doVoweltest gVowelTest
end
```

Movie Script38:WaitHandlers

```
on wait Howlong
  -- New Improved wait handler. doesn't reset timer
  -- every time it's called. be sure to "StartTimer in
  -- "on StartMovie"
  set x = the timer
  put x into oldtime -- stores time
  repeat while (oldtime + Howlong) > x
    nothing
    set x = the timer
  end repeat
end wait
```

```
on waitPlus Howlong, doWhat
  -- Same as above handler except that allows passing
  -- of a handler to be executed during the wait
  -- Handler must be a string
  set x = the timer
  put x into oldtime -- stores time
  repeat while (oldtime + Howlong) > x
    do doWhat -- must be a string
    set x = the timer
  end repeat
end wait
```

```
on IgnoreMouseDowns
  if the mousedown then dontpassevent
end
```

Movie Script39:spriteServices

```
on spritesOnList Spritelist
  --Takes a LIST of non-consecutive (or consecutive) channels
  -- and puppets them. Must be passed as a list []
  --turns global list gSpritesOnList off first and
  -- then turns on spritelist and makes Spritelist
  -- into gSpritesOnList
  global gSpritesOnList
  if count(gSpritesOnList) > 0 then
    repeat with thisSprite in gSpritesOnList
      puppetsprite (thisSprite, false)
    end repeat
    repeat with thisSprite in spritelist
      puppetsprite thisSprite, true
    end repeat
    set gSpritesOnList = spritelist
  else
    repeat with thisSprite in spritelist
      puppetsprite thisSprite, true
    end repeat
    set gSpritesOnList = spritelist
  end if
end

on spriteListOff
  -- turns off all sprites on Current gSpritesonList
  -- and re-initializes that global
  global gSpritesOnList
  if VoidP(gSpritesOnList) then exit
  repeat with thisSprite in gSpritesonList
    puppetsprite thisSprite, false
  end repeat
  set gSpritesOnList = []
end

on spriteson FirstSprite, LastSprite
  -- turns on sprites in consecutive channels from
  -- FirstSprite to LastSprite
  global gSpritesOnList
  if count(gSpritesOnList) > 0 then
    repeat with thisSprite in gSpritesOnList
      puppetsprite (thisSprite, false)
    end repeat
    set gSpritesOnList = []
    repeat with N = FirstSprite to LastSprite
      puppetsprite N, true
      add gSpritesonList, N
    end repeat
  else
    set gSpritesOnList = []
    repeat with N = FirstSprite to LastSprite
      puppetsprite N, true
      add gSpritesonList, N
    end repeat
  end if
end

on Spritesoff FirstSprite, LastSprite
  -- turns off sprites in consecutive channels from
```

```
-- FirstSprite to LastSprite  
repeat with N = FirstSprite to LastSprite  
  puppetsprite N, false  
end repeat  
end
```

Score Script40

```
on exitFrame  
  SetUserTimeout (600, "CVTestTimeout")  
end
```

Score Script41

```
on exitFrame  
  
end
```

Movie Script43

```
on VowelTestTimeout  
  global gVowelTest  
  doTimeout gVowelTest  
end  
  
on CVTestTimeout  
  global gCVTest  
  doTimeout gCVTest  
end
```

Score Script44

```
on exitFrame  
  go to the frame  
end
```

Script of Cast Member47

```
on mouseUp
    go to movie "Datatest"
end
```

Script of Cast Member51

```
on mouseUp
    go to frame "playagain"
end
```

Movie Script53

```
on SetUserTimeout howLong, doWhat
    global gUserTimeoutTime, gTimeoutHandler
    put the Ticks + howLong into gUserTimeoutTime
    put doWhat into gTimeoutHandler
end
```

```
on CheckUserTimeout
    global gUserTimeoutTime, gTimeoutHandler
    if the ticks < gUserTimeoutTime then
        exit
    else
        do gTimeoutHandler
        set gTimeoutHandler = empty
    end if
end
```

Script of Cast Member57

Score Script58

```
on mouseUp
  global gVowelTest
  repeatStimuli gVowelTest
  SetUserTimeOut (600,"vowelTestTimeOut")
end
```

Score Script59

```
on mouseUp
  global gCVtest
  repeatStimuli gCVtest
  SetUserTimeOut (600,"CVTestTimeOut")
end
```

Score Script60

```
on exitFrame
  if soundbusy(1) then
    go to the frame
  else
    end if
end
```

Score Script64

```
on exitFrame
  puppetsprite 10, true
  set the membernum of sprite 10 to 82
end
```

Score Script65

Score Script69

```
on exitFrame
  global gEgger
  putUpBrokenEgg gEgger, #Diff
end
```

Score Script75

```
on exitFrame
  global gHandCursor
  cursor gHandcursor
end
```

Score Script79

```
on exitFrame
  global gEgger
  putUpBrokenEgg gEgger, #same
end
```

Score Script92

```
on exitFrame
  global gRoundOver, gsoundFrame, gEgger
  if gRoundOver = false then
    putUpEgg gEgger
    go to frame gsoundFrame
  else
    putUpEgg gEgger
    go to frame "QuitOrGoOn"
    set gRoundOver = false
  end if
end
```

Score Script93

```
on exitFrame  
  global gEgger  
  putUpBrokenEgg gEgger, #diff  
end
```

Score Script94

```
on exitFrame  
  global gEgger  
  putUpBrokenEgg gEgger, #same  
end
```

Score Script95

Score Script96

```
on exitFrame
  -- new place to set up round
  -- needed to do here so we can poll the correct game object
  -- to play sample sounds during the instructions that follow.
  global gEgger, gRecordKeeper, gTheName, gLevel, gBounceNum
  init gEgger
  if objectP(gRecordKeeper) then
    put setUpRound (gRecordKeeper, gTheName, 3) into glevel
    put getGamePrefs (gRecordKeeper, gTheName, 2) into prefList

    if ListP(gLevel) then -- we are playing the second game
      set gBounceNum = getat(gLevel, 2)
      set gLevel = getat(gLevel, 1)
      if glevel = 115 then set glevel = 114
    end if
  else
    -- for authoring only!!!! if not starting from "starter" movie
    -- and hence no "gRecordKeeper" object, this will play the game at
    -- level 1. Change gLevel to any you might want.
    -- it will only play at that level. it won't save any new levels
    set gLevel = 1
    set gBounceNum = 0
  end if
  -- set gLevel = 31
  convertScoringLevel glevel, prefList -- uses glevel to figure out which sounds will be
  playing
end
```

Score Script97

Score Script98

```
on exitFrame
  global gEgger
  doEggBasket gEgger
end
```

Score Script99

```
on exitFrame  
  global gEgger  
  --Eg Basket trigger  
end
```

Score Script100

```
on exitFrame  
  global gEgger  
  set gEgger = new(script "egger")  
  sound stop 1  
  puppetSound 1.0  
  unloadmember  
  go to frame "intro"  
end
```

Score Script107

```
on mouseUp  
  global gCVtest  
  repeatStimuli gCVtest  
  SetUserTimeout (600, "CVTestTimeout")  
end
```

Score Script110

```
on exitFrame  
  if soundbusy(2) then  
    go to the frame  
  end if  
  
end
```

Score Script118

```
on mouseUp
  global qVowelTest.gpausestatus
  if gpausestatus = "resume" then exit
  repeatStimuli qVowelTest
  setUserTimeout (600, "vowelTestTimeout")
end
```

Score Script119

```
on mouseUp
  go to frame "black"
  go to movie "progress"
end
```

Movie Script120

```
on PlaySoundsinCast whichCast, whereStart
  -- plays every sound in the specified cast
  -- and posts name and number in message window.
  -- This allows quick check of sound and if the sound is looped
  global gWhereStart
  if not voidP(whereStart) then
    set y = wherestart
  else
    if not voidP(gWhereStart) then
      set y = gWhereStart
    else
      set y = 1
    end if
  end if
  repeat with x = y to the number of members of castlib whichCast
    if the type of member x of castlib whichCast = #sound then
      put x && the name of member x of castlib whichCast
      puppetsound member x of castlib whichCast
      set gWherestart = x-1
      updatestage
      repeat while soundBusy(1)
        if mouseDown() = true then
          puppetsound 0
          exit
          return 0
        end if
      end repeat
      wait 5 -- change here for longer pause between sounds
    end if
  end repeat
end
```

Score Script123

Score Script124

Score Script125

Score Script126

on mouseUp

set the visible of sprite 45 to false
set the visible of sprite 46 to false
set the visible of sprite 47 to false
sound stop 1
sound stop 2
puppetsound 0
go to marker(0)
continue

end

Score Script127

on mouseUp

set the visible of sprite 45 to false
set the visible of sprite 46 to false
set the visible of sprite 47 to false
sound stop 1
sound stop 2
puppetsound 0
go to marker(0)
continue

end

Score Script128

```
on mouseup  
  dontpasseevent  
  
end
```

Score Script129

```
on mouseUp  
  go to frame "playagain"  
end
```

Score Script131

```
on mouseup  
  global gpausestatus  
  set gpausestatus = "resume"  
  set the visible of sprite 46 to false  
  go to marker (0)  
  pause  
end
```

Score Script132

```
on mouseup  
  global gpausestatus  
  set gpausestatus = "resume"  
  set the visible of sprite 46 to false  
  -- go to marker (0)  
  pause  
end
```

Script of Cast Member136

```
on mouseUp
  sound stop 1
  sound stop 2
  puppetSound 0
  repeat with x = 1 to 48
    puppetSprite x, false
    set the visible of sprite x to true
  end repeat
  cursor 4
  go to frame "black"
  go to movie "progress"
end
```

Script of Cast Member137:pause.pict

```
--on mouseup
--  -- puppetSprite 45, true
--  -- set the member of sprite 45 to member "resume.pict"
--  -- updateStage
--  -- pause
--  set the visible of sprite 46 to false
--  pause
--end
```

Script of Cast Member138:resume.pict

```
on mouseUp
  global gpausestatus
  set gpausestatus = "pause"
  --set the member of sprite 45 to member "pause.pict"
  -- update stage
  -- puppet sprite 45, false
  set the visible of sprite 45 to false
  set the visible of sprite 46 to false
  set the visible of sprite 47 to false
  sound stop 1
  sound stop 2
  puppet sound 0
  -- go to marker (0)
  continue
end
```

Script of Cast Member139

on mouseUp
 global gpausestatus

 if the visible of sprite 47 = false then
 set the visible of sprite 45 to true
 set the visible of sprite 47 to true
 if gpausestatus = "pause" then
 set the visible of sprite 46 to true
 end if

 else
 set the visible of sprite 47 to false
 set the visible of sprite 45 to false

 set the visible of sprite 46 to false
 end if

 updatestage
 -- puppetsprite 45, true

end

Score Script142

```
on mouseUp
  global gpausestatus
  set gpausestatus = "pause"
  --set the member of sprite 45 to member "pause.pict"
  -- updatestage
  -- puppetsprite 45.false
  set the visible of sprite 45 to false
  set the visible of sprite 46 to false
  set the visible of sprite 47 to false
  sound stop 1
  sound stop 2
  puppetsound 0
  -- go to marker(0)
  SetUserTimeout (600,"CVTestTimeout")
  continue
end
```

Score Script144

```
on exitFrame
  if soundbusy(2) then
    go to the frame
  else
    repeat with x = 1 to 48
      puppetsprite x.false
    end repeat
  end if
end
```

Score Script145

```
on mouseUp
  go to frame "playagain"
end
```

Score Script146

```
on mouseUp
  cursor 4
  go to frame "seeya"
end
```

Script of Cast Member147

```
on mouseUp
  go to frame "black"
  go to movie "progress"
end
```

Script of Cast Member148

```
on mouseUp
  go to frame "playagain"
end
```

Score Script150

```
on exitFrame
  -- report scores here, after farmer says his piece
  global gSoundFrame, gVowelTest, gCVtest
  if soundbusy(2) then
    go to the frame
  else
    case (gSoundFrame) of
      (label("vowelSound")) : doEndOfRound gVowelTest
      (label ("CvSound")): doEndOfRound gCVtest
    end case
  end if
end
```

Score Script154

on exitFrame
 cursor: 4
 go to movie "progress"
end

Script of Cast Member155: no PCT

Score Script158

on exitFrame
 go to the frame

end

Score Script159

on exitFrame
 if soundbusy(2) then
 go to the frame
 end if

end

Score Script161

on exitFrame
 playSampleSounds #same
end

Score Script162

```
on exitFrame  
  playSampleSounds #Diff  
end
```

Score Script163

Movie Script164

```
on playSampleSounds whichSounds  
  -- uses gLevel to figure out which game is being played  
  -- then calls that game's object to play sample sounds  
  -- and passes on the parameters (either #Diff or #same)  
  -- to inform the object which sounds to play  
  global gCVtest, gVowelTest, gLevel  
  if gLevel <= 30 then  
    doSampleSounds (gVowelTest, whichSounds)  
  else  
    doSampleSounds (gCVtest, whichSounds)  
  end if  
end
```

Score Script165

Score Script166

```
on mouseUp  
  -- this script is here to  
  -- keep the cursor from flickering during the animated  
  -- sequences. Don't delete please.  
end
```

Score Script167

```
on exitFrame
  repeat with x = 45 to 47
    set the visible of sprite x to false
  end repeat
end
```

Script of Cast Member168

```
on mouseUp
  sound stop 1
  sound stop 2
  puppetsound 0
  repeat with x = 1 to 48
    puppetsprite x, false
    set the visible of sprite x to true
  end repeat

  go to frame "black"
  go to movie "progress"
end
```

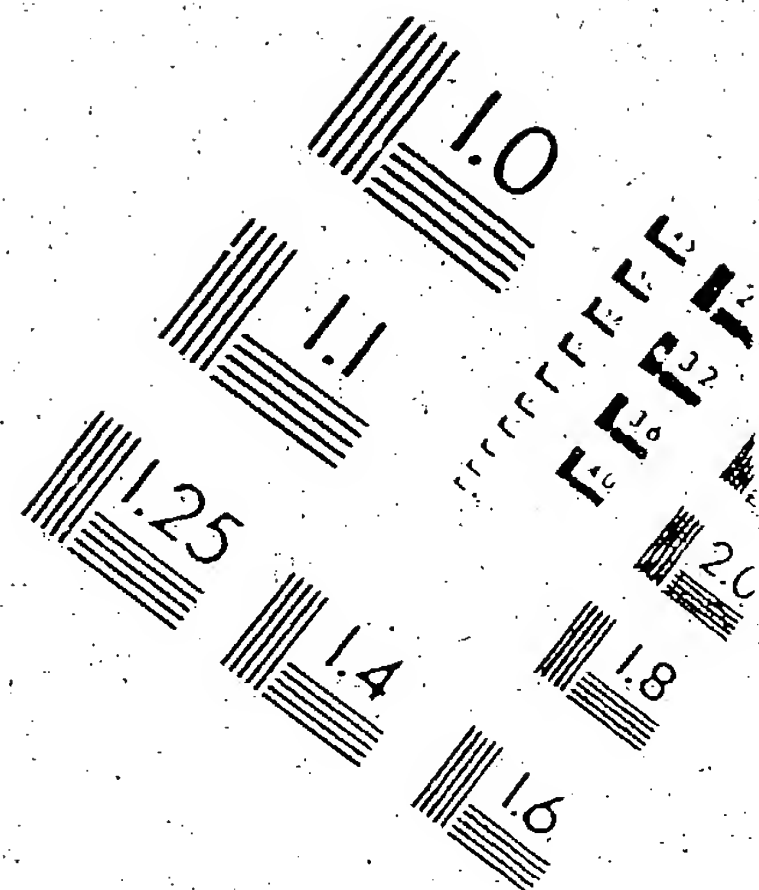
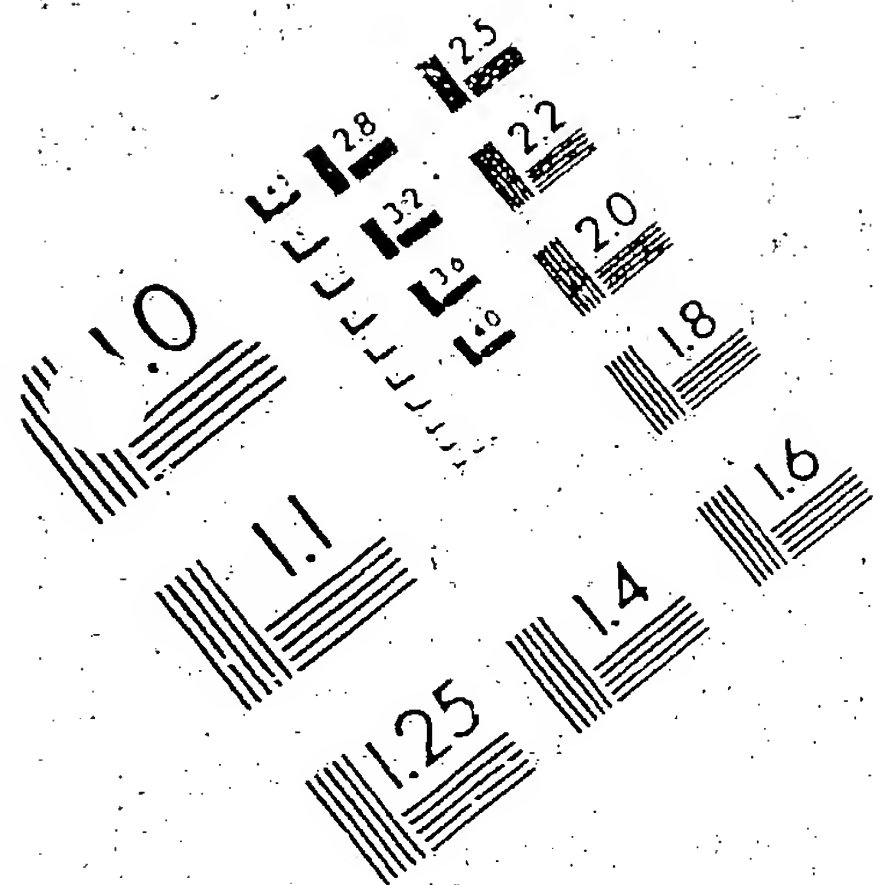
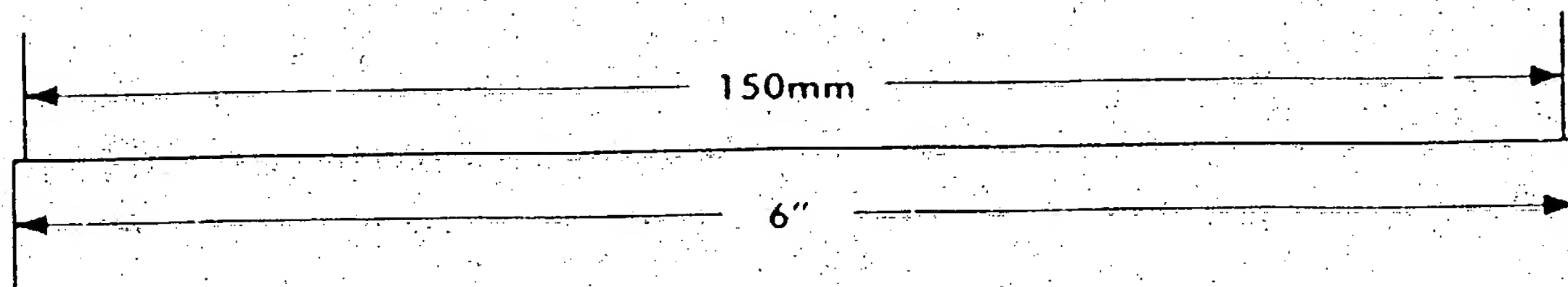
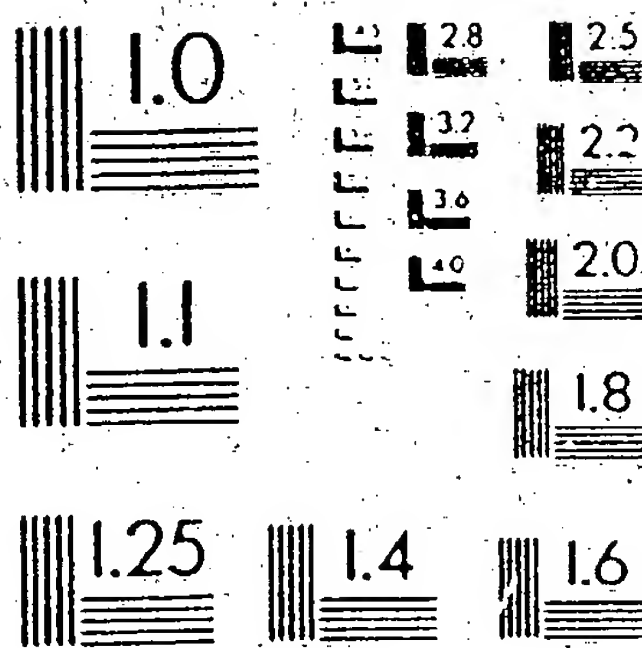
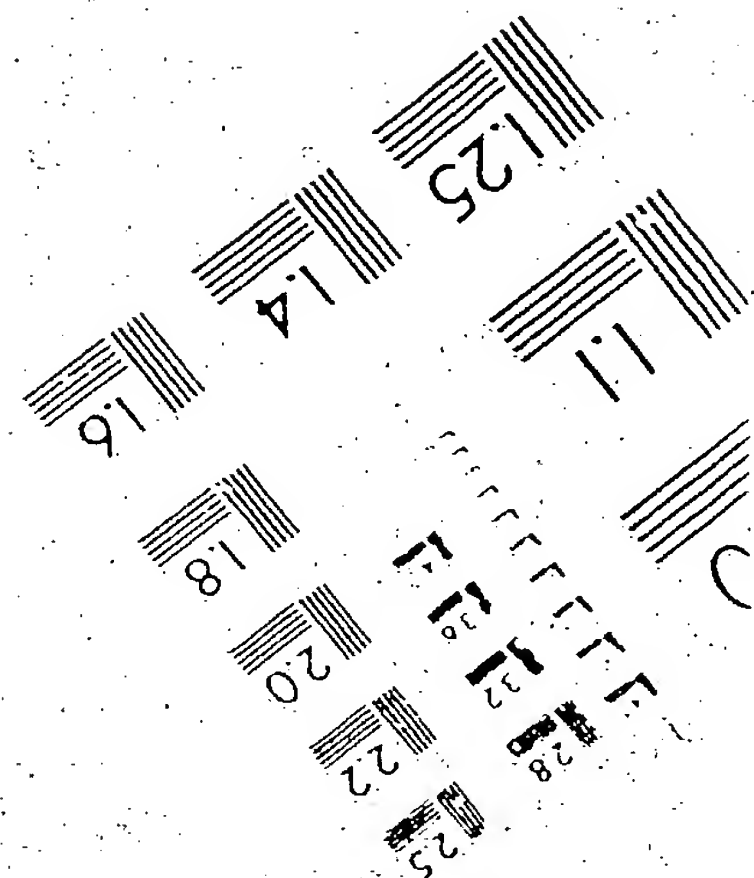
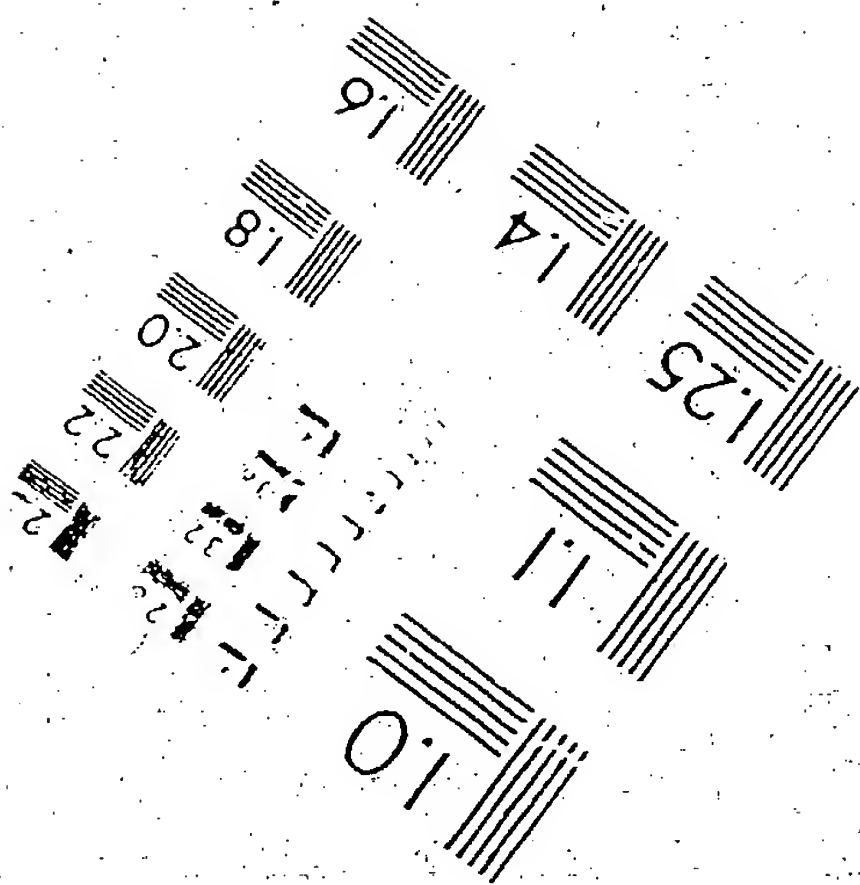


IMAGE EVALUATION
TEST TARGET QA-3



APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone 716 482 0300
Fax 716 288 5989



Script of Cast Member169

```
on mouseUp
  global gpausestatus

  if the visible of sprite 47 = false then
    set the visible of sprite 45 to true
    set the visible of sprite 47 to true
    if gpausestatus = "pause" then
      set the visible of sprite 46 to true
    end if
  else
    set the visible of sprite 47 to false
    set the visible of sprite 45 to false

    set the visible of sprite 46 to false
  end if

  updatestage
  -- puppetsprite 45, true
end
```

Score Script170

Score Script171

Score Script172

```
on exitFrame
  if soundbusy(2) then
    go to the frame
  end if
end
```

Score Script173

```
on exitFrame
  global gVowelTest
  if the disableReplayButton of gVowelTest = false then
    -- show the replay Button
    puppetSprite 25, true
    puppetSprite 26, true
    set the loc of sprite 25 = point(611,25)
    set the loc of sprite 26 = point(580,-1)
    updatestage
  end if
end
```

Score Script174

```
on exitFrame
  global gCVTest
  if the disableReplayButton of gCVTest = false then
    -- show the replay Button
    puppetSprite 25, true
    puppetSprite 26, true
    set the loc of sprite 25 = point(611,25)
    set the loc of sprite 26 = point(580,-1)
    updatestage
  end if
end
```

Score Script175

```
on exitFrame
  global gVowelTest, gCVTest
  if the skipYesOrNo of gVowelTest or the skipYesOrNo of gCVTest = true then
    repeat with x = 45 to 47
      set the visible of sprite x to false
    end repeat
    go to frame "playAgain"
  end if
end
```

Score Script176

```
on exitFrame
  preload the frame, the frame + 3
  -- preloadmember "same"
  -- preloadmember 2
end
```

Parent Script201:CVTestProctor 4.9

Property TestPairList, ancestor, myHandlers, currentLevel, DiffList, SameList
property roundScoreList, SoundGroup, Answer, NumRightInARow, NumWrongInARow
Property LevelToSave, PairToPlay, timeoutNum, TimeOutAnswer

on x-----Public Handlers -----
 -- I'm a separator
end

on new me
 global gRecordKeeper
 if ObjectP(gRecordKeeper) then
 set the ancestor of me to gRecordKeeper
 set myHandlers = 0
 set myHandlers = GetMyHandlers(me)
 end if
 return Me
end

on startNewRound me, whichLevel, whichsoundGroup
 -- Pub.
 -- sets up lists of soundpairs for current group of levels
 -- "WhichSoundGroup" is a string which all the sounds for this
 -- group share with only numerical suffixes changing
 -- the soundGroup property never changes during a round
 global gRoundOver
 setUpTestPairList me
 set RoundScoreList = [:]
 set soundgroup = whichsoundGroup
 initDiffSameLists me
 set numRightInARow = 0
 set NumWrongInARow = 0
 set currentlevel = whichLevel
 set timeoutNum = 0
 set TimeOutanswer = empty
 set gRoundOver = false
 put soundGroup
end

on doSampleSounds me, whichSounds
 -- pub.
 -- called from the score (ultimately), plays a pair of
 -- same sounds if "whichSounds" = #same or different sounds
 -- if "whichSounds" = #diff. Sounds are pulled from the
 -- current sound group
 if whichsounds = #same then
 set sound1 = soundGroup&1
 set samplePair = [sound1,sound1]
 else
 set sound1 = soundGroup&1
 set sound2 = soundgroup&9
 set samplePair = [sound1,sound2]
 end if
 set firstSound = getat(samplePair,1)
 set secondSound = getat(samplePair,2)
 puppetsound firstSound
 updatestage
 repeat while soundBusy(1)

```

    nothing
end repeat
puppetsound 0
wait 30
puppetsound secondSound
updatestage
repeat while soundBusy(1)
    nothing
end repeat
puppetsound 0
end

```

```

on DoCVTest me
    -- pub.
    -- this is handler to play test
    -- it plays the two sounds and sets the property "answer" according to
    -- whether the answer is same or different. These options are passed as
    -- symbols, ie. #same or #diff
    set level = (currentlevel-30)mod(7)
    if level = 0 then set Level = 7
    set PlaysLeft = count(testPairList)
    if PlaysLeft < 1 then exit
    set NextPair = getAt(testPairList, playsLeft)
    deleteAt (testPairList, playsLeft)
    if nextPair = #Diff then
        set pairList = getprop(difflist, Level)
        set pair = random(count(PairList))
        set PairToPlay = getAt(PairList, Pair)
        set Answer = #Diff
    else
        set pairList = getprop(samelist, Level)
        set pair = random(count(PairList))
        set PairToPlay = getAt(PairList, Pair)
        set answer = #same
    end if
    put answer
    set sound1 = getAt(PairToPlay, 1)
    puppetsound member sound1
    updatestage
    wait 30
    set sound2 = getAt(PairToPlay, 2)
    puppetsound member sound2
    updatestage
    put the name of member sound1
    put the name of member sound2
    put "current level = " & currentLevel
end

```

```

on RepeatStimuli me
    -- pub.
    -- replays two current sounds if
    -- user needs to
    set sound1 = getAt(PairToPlay, 1)
    set sound2 = getAt(PairToPlay, 2)
    puppetsound member sound1
    updatestage
    wait 30
    puppetsound member sound2
    updatestage
end

```

on CheckAnswer me, whichAnswer

```
-- this handler should be called from the
-- choice buttons, ie the hen pairs on screen.
-- it evaluates the answer given by the user and keeps a property list
-- accordingly to report to the RecordKeeper. The answers must be passed
-- as symbols #Same or #diff. The list being kept is a property list
-- with the current level as the property, the value is itself a list
-- with 4 items, the first 2 are for scores on different sounds, the
-- last 2 are for scores on same sounds.
-- This handler also keeps track of how many right or wrong in a row
-- the user answers during a round. If 4 right answers in a row then
-- the level is raised by 1, if 2 wrong in a row it drops by 1.
```

global gRoundOver, gWhichEgg

```
if whichAnswer = answer then -- Right!!
```

```
  set gWhichEgg = "Correct Egg"
```

```
  set NumRightInARow = NumRightInARow + 1
```

```
  set NumWrongInARow = 0
```

```
  if answer = #diff then
```

```
    go to frame "Diff correct"
```

```
    set AnsList = getaProp(roundscoreList, currentlevel)
```

```
    if voidP(AnsList) then -- no score at this level yet
```

```
      set AnsList = [1,1,0,0]
```

```
      addProp roundscorelist, currentLevel, ansList
```

```
    else
```

```
      set NumPlays = getat (ansList, 1)
```

```
      set NumPlays = numplays + 1
```

```
      set NumRight = getat(ansList, 2)
```

```
      set Numright = numright + 1
```

```
      setAt AnsList, 1, numPlays
```

```
      setAt AnsList, 2, numRight
```

```
      setaProp RoundscoreList, CurrentLevel, anslist
```

```
    end if
```

```
  else -- answer was #same
```

```
    go to frame "Same Correct"
```

```
    set AnsList = getaProp(roundscoreList, currentlevel)
```

```
    if voidP(AnsList) then -- no score at this level yet
```

```
      set AnsList = [0,0,1,1]
```

```
      addProp roundscorelist, currentLevel, ansList
```

```
    else
```

```
      set NumPlays = getat (ansList, 3)
```

```
      set NumPlays = numplays + 1
```

```
      set NumRight = getat(ansList, 4)
```

```
      set Numright = numright + 1
```

```
      setAt AnsList, 3, numPlays
```

```
      setAt AnsList, 4, numRight
```

```
      setaProp RoundscoreList, CurrentLevel, anslist
```

```
    end if
```

```
  end if
```

```
else -- wrong!!
```

```
  set gWhichEgg = "wrong Egg"
```

```
  set NumRightInARow = 0
```

```
  set NumWrongInARow = NumwrongInARow + 1
```

```
  if answer = #diff then
```

```
    go to frame "same Wrong"
```

```
    set AnsList = getaProp(roundscoreList, currentlevel)
```

```
    if voidP(AnsList) then -- no score at this level yet
```

```
      set AnsList = [1,0,0,0]
```

```

    addProp roundscorelist, currentLevel, ansList
else
    set NumPlays = getat (ansList, 1)
    set NumPlays = numplays + 1
    setAt AnsList, 1, numPlays
    setaProp RoundscoreList, CurrentLevel, anslist
end if

else
    go to frame "diff Wrong"
    set AnsList = getaProp(roundscoreList,currentlevel)
    if voidP(AnsList) then -- no score at this level yet
        set AnsList = {0,0,1,0}
        addProp roundscorelist, currentLevel, ansList
    else
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        setAt AnsList, 3, numPlays
        setaProp RoundscoreList, CurrentLevel, anslist
    end if
end if

end if
if numrightinarow > 3 then
    if (currentlevel-30)mod(7) = 0 then -- we are at level 7 within a group
        set currentlevel = currentlevel
        set LeveltoSave = CurrentLevel + 1
    else
        set currentLevel = currentlevel + 1
        set LeveltoSave = currentLevel
    end if
    set NumrightInarow = 0
end if
if numwronginarow > 1 then
    if (currentlevel-30)mod(7) = 1 then
        set currentLevel = currentlevel
        set leveltoSave = currentlevel
    else
        set currentlevel = currentlevel - 1
        set leveltoSave = currentlevel
    end if
    set numWronginArow = 0
end if
if count(testPairList) < 1 then
    set gRoundOver = true
    exit
end if

end

```

```

on doTimeOut me
    global gWhichEgg, gRoundOver
    if count(testPairList) < 1 then -- end of round so set flag
        set gRoundOver = true
    end if

    if TimeOutNum = 1 then -- second timeOut in a row so bailout
        go to frame "playAgain?"
    end if

```

```

if count(roundscoreList) > 0 then
  -- user played some in this round so report scores
  doEndOfRound me
end if
set timeoutNum = 0
set TimeoutAnswer = empty
else
  -- first adjust current level for wrong answer
  set NumWrongInARow = NumWrongInARow + 1
  set NumRightInARow = 0
  if numWrongInARow > 1 then
    if (currentLevel-30) mod(7) = 1 then
      set currentLevel = currentLevel
      set levelToSave = currentLevel
    else
      set currentLevel = currentLevel - 1
      set levelToSave = currentLevel
    end if
    set numWrongInARow = 0
  end if
  if gRoundOver = false then -- not last egg
    -- do egg drop and store scores to report
    -- if user plays again. else bailout above happens
    set gWhichEgg = "Wrong egg"
    set timeoutNum = TimeoutNum + 1
    set TimeoutAnswer = answer
    if answer = #Diff then
      go to frame "Diff Drop"
    else
      go to frame "same Drop"
    end if
  else -- timeout on last egg so do scoring here
    set gWhichEgg = "Wrong egg"
    set AnsList = getaProp(roundscoreList,currentLevel)
    if voidP(ansList) then set AnsList = [0,0,0,0] -- no scores yet at this level
    if answer = #Diff then
      go to frame "Diff Drop"
      set NumPlays = getaT(ansList, 1)
      set NumPlays = numPlays + 1
      setAt AnsList, 1, numPlays
    else
      go to frame "same Drop"
      set NumPlays = getaT(ansList, 3)
      set NumPlays = numPlays + 1
      setAt AnsList, 3, numPlays
    end if
    setaProp roundscoreList, currentLevel, ansList
  end if
end if
end

```

```

on CheckForTimeout me
  -- call from "answer chickens" on stage
  -- resets timeout and reports timeout score
  if TimeoutNum > 0 then -- this is second Timeout
    set timeoutNum = 0 -- reset counter

    set AnsList = getaProp(roundscoreList,currentLevel)
    -- retrieve list to report score from timeout egg
    if voidP(ansList) then set AnsList = [0,0,0,0] -- no scores yet at this level
  end if
end

```



```

    if TimeOutanswer = #diff then
        set NumPlays = getat (ansList, 1)
        set NumPlays = numplays + 1
        setAt AnsList, 1, numPlays
    else
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        setAt AnsList, 3, numPlays
    end if
    setaProp roundScoreList, currentLevel, anslist
end if
end

on doEndOfRound me
    if not objectP(gRecordKeeper) then exit
    if voidP(leveltoSave) then set leveltosave = currentLevel
    if leveltosave > 115 then set leveltosave = 115
    saveroundscores(me,roundscorelist,leveltosave)
end

on xx-----Private Handlers-----
    -- i'm a separator
end

on InitDiffSameLists me
    set s1 = soundgroup&1
    set s1 = the number of member s1
    set s2 = soundgroup&2
    set s2 = the Number of member s2
    set s3 = soundgroup&3
    set s3 = the Number of member s3
    set s4 = soundgroup&4
    set s4 = the Number of member s4
    set s5 = soundgroup&5
    set s5 = the Number of member s5
    set s6 = soundgroup&6
    set s6 = the Number of member s6
    set s7 = soundgroup&7
    set s7 = the Number of member s7
    set s8 = soundgroup&8
    set s8 = the Number of member s8
    set s9 = soundgroup&9
    set s9 = the Number of member s9

    set DiffList = [:]
    set SameList = [:]
    repeat with x = 1 to 7
        case (x) of
            1: addProp DiffList ,1, [[s1,s9],[s9,s1]]
               addProp samelist ,1, [[s1,s1],[s9,s9]]
            2: addProp DiffList ,2, [[s1,s8],[s8,s1],[s2,s9],[s9,s2]]
               addProp samelist ,2, [[s1,s1],[s2,s2],[s8,s8],[s9,s9]]
            3: addProp DiffList ,3, [[s1,s7],[s7,s1],[s2,s8],[s8,s2],[s3,s9],[s9,s3]]
               addProp samelist ,3, [[s1,s1],[s2,s2],[s3,s3],[s7,s7],[s8,s8],[s9,s9]]
            4: addProp DiffList ,4,
               [[s1,s6],[s6,s1],[s2,s7],[s7,s2],[s3,s8],[s8,s3],[s4,s9],[s9,s4]]
               addProp samelist ,4,
               [[s1,s1],[s2,s2],[s3,s3],[s4,s4],[s6,s6],[s7,s7],[s8,s8],[s9,s9]]
            5: addProp DiffList ,5, [[s2,s6],[s6,s2],[s3,s7],[s7,s3],[s4,s8],[s8,s4]]

```

```

        addProp samelist ,5, [[s2,s2],[s3,s3],[s4,s4],[s6,s6],[s7,s7],[s8,s8]]
6: addProp DiffList ,6, [[s3,s6],[s6,s3],[s4,s7],[s7,s4]]
        addProp samelist ,6, [[s3,s3],[s4,s4],[s6,s6],[s7,s7]]
7: addProp DiffList ,7, [[s4,s6],[s6,s4]]
        addProp samelist ,7, [[s4,s4],[s6,s6]]
    end case
end repeat
end

```

```

on setUpTestPairList me
    set testpairlist to []
    repeat with x = 1 to 10
        append testpairlist , #Diff
    end repeat
    set TempList = [1,2,3,4,5,6,7,8,9,10]
    repeat with x = 1 to 6
        set y = count(tempList)
        deleteat tempList, random(y)
    end repeat
    repeat with pos in tempList
        setAt(testpairlist,pos, #same)
    end repeat
    put testpairlist
end

```

```

on xxx-----Testing Handlers-----
    -- i'm a separator
    nothing
end

```

```

on showhandlers me
    put myhandlers
end

```

```

on showProps me
    -- testing
    -- puts list of properties and their current values in message window
    set PropNum = count(me)
    repeat with x = 1 to PropNum
        set prop = 0
        set thisProp = getpropat(me, x)
        if thisProp = #myHandlers then next repeat
        put (string (getpropat(me, x))) &&"="&& getaProp(me, thisProp) into prop
        put prop
    end repeat
end

```


Parent Script202:vowelTest 4.9

```
property TestPairList, currentLevel, RoundList, ancestor, myHandlers, answer
property pairToPlay, TimeOutNum, TimeOutanswer, sampleSoundList
```

```
on x-----Public Handlers-----
  -- I'm a separator
end
```

```
on new me
  global gRecordKeeper
  if objectP(gRecordKeeper) then
    set the ancestor of me to gRecordKeeper
    set myHandlers = 0
    set myHandlers = GetMyHandlers(me)
  end if
  return me
end
```

```
on startNewRound me, level
  global gRoundOver
  set CurrentLevel = level
  setUpVowelList me, currentLevel
  set RoundList = [:]
  addProp (roundList, currentLevel, [0,0,0,0])
  set TimeOutNum = 0
  set gRoundOver = false
end
```

```
on doSampleSounds me, whichSounds
  -- pub
  -- called from the score (ultimately), plays two pairs of
  -- same sounds if "whichSounds" = #same or different sounds
  -- if "whichSounds" = #diff. Sounds are stored in two lists in property
  -- SampleSoundList, first sublist are same, second are different
  if whichSounds = #same then
    set samplePairs = [getat(sampleSoundList,1),getat(sampleSoundList,2)]
  else
    set samplePairs = [getat(sampleSoundList,3),getat(sampleSoundList,4)]
  end if
  repeat with samplePair in samplePairs
    set firstSound = getat(samplePair,1)
    set secondSound = getat(samplePair,2)
    puppetsound member firstSound
    updatestage
    repeat while soundBusy(1)
      nothing
    end repeat
    puppetsound 0
    wait 30
    puppetsound member secondSound
    updatestage
    repeat while soundBusy(1)
      nothing
    end repeat
    puppetsound 0
    wait 90
  end repeat
end
```

```

on doVowelTest me
  set PlaysLeft = count(testPairList)
  if PlaysLeft < 1 then exit
  set pairToPlay = getat(testPairList, playsLeft)
  deleteAt (testPairList, playsLeft)
  set sound1 = getat(pairToPlay, 1)
  set sound2 = getat(pairToPlay, 2)
  if sound1 = sound2 then
    set answer = #same
  else
    set answer = #diff
  end if
  puppetSound sound1
  updateStage
  wait 30
  puppetSound sound2
  updateStage
  put answer
  put the name of member sound1
  put the name of member sound2
end

```

```

on RepeatStimuli me
  set sound1 = getat(pairToPlay, 1)
  set sound2 = getat(pairToPlay, 2)
  puppetSound sound1
  updateStage
  wait 30
  puppetSound sound2
  updateStage
end

```

```

on checkAnswer me, whichAnswer
  global gRoundOver, gWhichEgg
  set AnsList = getaProp(roundList, currentlevel)
  if whichAnswer = answer then -- right
    set gWhichEgg = "Correct Egg"
    if answer = #diff then
      set NumPlays = getat (ansList, 1)
      set NumPlays = numplays + 1
      set NumRight = getAt(ansList, 2)
      set Numright = numright + 1
      setAt AnsList, 1, numPlays
      setAt AnsList, 2, numRight
      go to frame "diff Correct"
    else -- answer was #same
      set NumPlays = getat (ansList, 3)
      set NumPlays = numplays + 1
      set NumRight = getAt(ansList, 4)
      set Numright = numright + 1
      setAt AnsList, 3, numPlays
      setAt AnsList, 4, numRight
      go to frame "same Correct"
    end if
  else -- wrong!!
    set gWhichEgg = "wrong Egg"
    if answer = #diff then
      set NumPlays = getat (ansList, 1)
      set NumPlays = numplays + 1

```

```

    setAt AnsList, 1, numPlays
    go to frame "same wrong"
else
    set NumPlays = getAt (ansList, 3)
    set NumPlays = numPlays + 1
    setAt AnsList, 3, numPlays
    go to frame "diff wrong"

end if
end if
setAProp RoundList, CurrentLevel, anslist
if count(testPairList) < 1 then

    set gRoundOver = true
    exit
end if
end

on doEndOfRound me
    if not objectP(gRecordKeeper) then exit
    set scores = getProp(roundList, currentLevel)
    if (getAt(scores, 2) >= 5) and (getAt(scores, 4) = 4) then
        set levelToSave = currentLevel + 1
    else
        set levelToSave = currentLevel
    end if
    SaveRoundScores (me, roundlist, levelToSave)
end

on doTimeout me
    global gWhichEgg, gRoundOver
    if count(testPairList) < 1 then
        set gRoundOver = true
    end if
    if TimeoutNum = 1 then
        go to frame "PlayAgain?"
        set AnsList = getAProp(roundList, currentlevel)
        if getAt(anslist, 1) > 0 or getAt(anslist, 3) > 0 then
            -- user played some in this round
            doEndOfRound me
        end if
        set the timeoutscript = empty
        set timeoutNum = 0
        set TimeoutScore = empty
    else
        if gRoundOver = false then -- not last egg
            set gWhichEgg = "Wrong egg"
            set timeoutNum = TimeoutNum + 1
            Set Timeoutanswer = answer
            if answer = #Diff then
                go to frame "Diff Drop"
            else
                go to frame "same Drop"
            end if
        else -- timeout on last egg!!
            set gWhichEgg = "Wrong egg"
            set AnsList = getAProp(roundList, currentlevel)
            if answer = #diff then
                go to frame "Diff Drop"
            end if
        end if
    end if
end

```

```

        set NumPlays = getat (ansList, 1)
        set NumPlays = numplays + 1
        setAt AnsList, 1, numPlays
    else
        go to frame "same.Drop"
        set NumPlays = getat (ansList, 3)
        set NumPlays = numplays + 1
        setAt AnsList, 3, numPlays
    end if
end if
end if
end

```

```

on CheckForTimeOut me
    -- call from "answer chickens" on stage
    -- resets timeOut and reports timeoutScore
    if TimeOutNum > 0 then
        set timeOutNum = 0
        set AnsList = getaProp(roundList,currentlevel)
        if TimeOutanswer = #diff then
            set NumPlays = getat (ansList, 1)
            set NumPlays = numplays + 1
            setAt AnsList, 1, numPlays
        else
            set NumPlays = getat (ansList, 3)
            set NumPlays = numplays + 1
            setAt AnsList, 3, numPlays
        end if
    end if
end

```

```

on xx-----Private Handlers-----
    -- i'm a separator
end

```

```

on setUpVowelList me, whichLevel
    -- sets up list of 10 lists. each sublist contains two items,
    -- the names of sounds from a global list "gVowelSounds"
    -- these sounds are either paired with themselves or are
    -- paired with the other sound. As per instruction, the handler sets up
    -- the ten list randomly with 6 lists of "different" pairs and
    -- 4 lists of "same" pairs
    global gVowelSounds
    if whichLevel > count(gVowelSounds) or whichLevel < 1 then
        alert "There are no vowel sounds at that level. Check your lingo."
        abort
    end if
    set soundlist = getat(gvowelsounds, whichLevel)
    repeat with x = 1 to 2 -- change names to cast numbers
        set Sound = getat(soundList, x)
        set sound = the member of member sound
        setat soundlist, x, sound
    end repeat

```

```

set xx = [getat(soundlist,1),getat(soundlist,1)]
set xy = [getat(soundlist,1),getat(soundlist,2)]
set yy = [getat(soundlist,2),getat(soundlist,2)]
set yx = [getat(soundlist,2),getat(soundlist,1)]
set VowelPairList = []
set sameList = [xx,yy]
set diffList = [xy,yx]

set sampleSoundList = [xx,yy,xy,yx]
-- for use during "doSampleSounds" handler
-- first two items are lists of two same sounds
-- second two are lists of two different sounds

repeat with x = 1 to 10
  append VowelPairList, getat(diffList, random(2))
end repeat
set TempList = [1,2,3,4,5,6,7,8,9,10]
repeat with x = 1 to 6
  set y = count(tempList)
  deleteat tempList, random(y)
end repeat
repeat with pos in tempList
  setAt(VowelPairList,pos, getat(sameList,random(2)))
end repeat
set testPairList = vowelPairList
end

on xxx-----Testing Handlers-----
  -- i'm a separator
  nothing
end

on showhandlers me
  put myhandlers
end

on showProps me
  -- testing
  -- puts list of properties and their current values in message window
  set PropNum = count(me)
  repeat with x = 1 to PropNum
    set prop = 0
    set thisProp = getpropat(me, x)
    if thisProp = #myHandlers then next repeat
    put (string (getpropat(me, x))) &&"=" && getaProp(me,thisProp) into prop
    put prop
  end repeat
end

```

Script of Cast Member2:Age